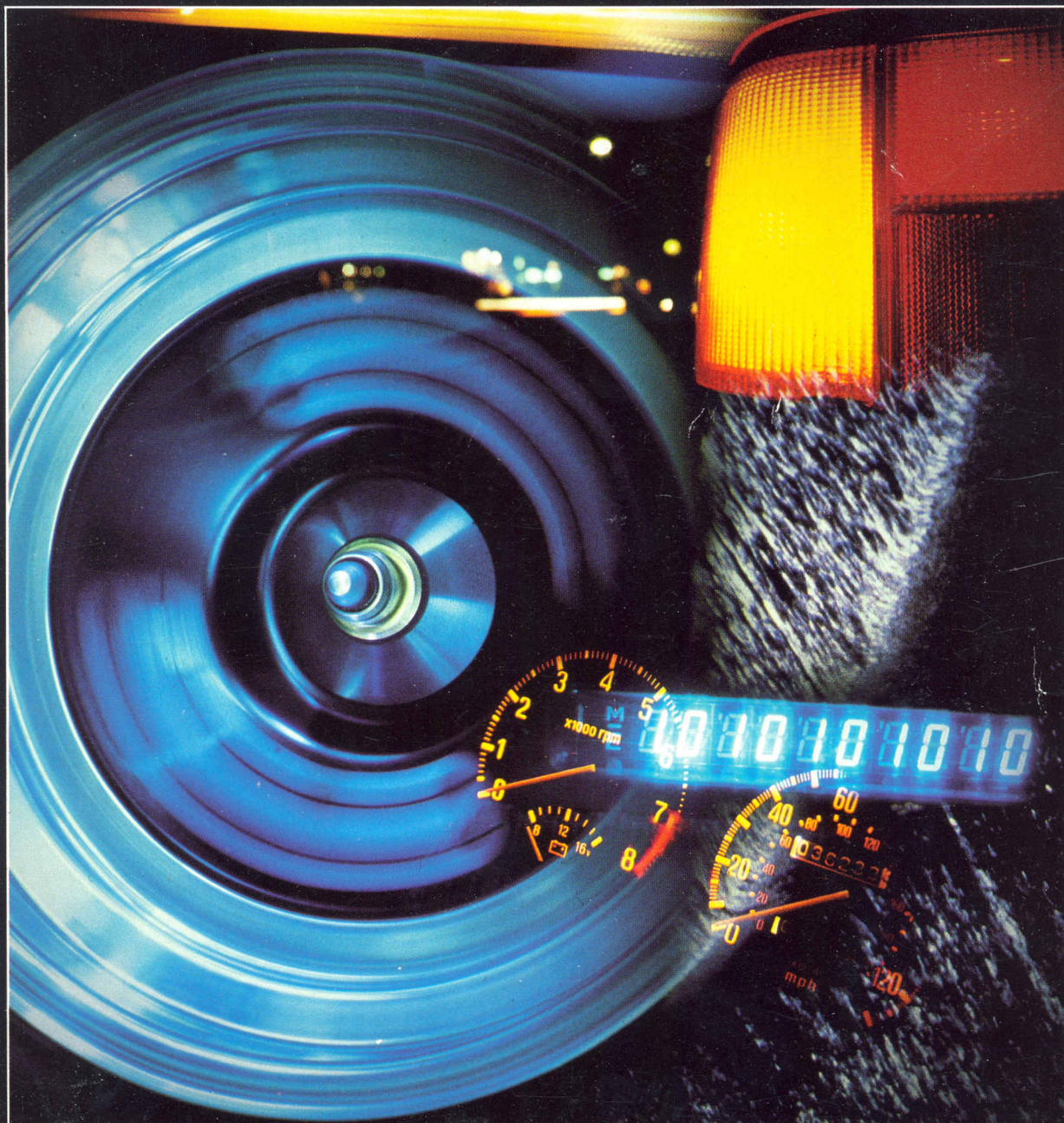




Automotive Handbook



Order Number: 231792-002

LITERATURE

To order Intel literature write or call:

Intel Literature Sales
P.O. Box 58130
Santa Clara, CA 95052-8130

Toll Free Number:
(800) 548-4725*

Use the order blank on the facing page or call our Toll Free Number listed above to order literature. Remember to add your local sales tax and a 10% postage charge for U.S. and Canada customers, 20% for customers outside the U.S. Prices are subject to change.

1988 HANDBOOKS

Product line handbooks contain data sheets, application notes, article reprints and other design information.

NAME	ORDER NUMBER	**PRICE IN U.S. DOLLARS
COMPLETE SET OF 8 HANDBOOKS Save \$50.00 off the retail price of \$175.00	231003	\$125.00
AUTOMOTIVE HANDBOOK (Not included in handbook Set)	231792	\$20.00
COMPONENTS QUALITY/RELIABILITY HANDBOOK (Available in July)	210997	\$20.00
EMBEDDED CONTROLLER HANDBOOK (2 Volume Set)	210918	\$23.00
MEMORY COMPONENTS HANDBOOK	210830	\$18.00
MICROCOMMUNICATIONS HANDBOOK	231658	\$22.00
MICROPROCESSOR AND PERIPHERAL HANDBOOK (2 Volume Set)	230843	\$25.00
MILITARY HANDBOOK (Not included in handbook Set)	210461	\$18.00
OEM BOARDS AND SYSTEMS HANDBOOK	280407	\$18.00
PROGRAMMABLE LOGIC HANDBOOK	296083	\$18.00
SYSTEMS QUALITY/RELIABILITY HANDBOOK	231762	\$20.00
PRODUCT GUIDE Overview of Intel's complete product lines	210846	N/C
DEVELOPMENT TOOLS CATALOG	280199	N/C
INTEL PACKAGING OUTLINES AND DIMENSIONS Packaging types, number of leads, etc.	231369	N/C
LITERATURE PRICE LIST List of Intel Literature	210620	N/C

*Good in the U.S. and Canada

**These prices are for the U.S. and Canada only. In Europe and other international locations, please contact your local Intel Sales Office or Distributor for literature prices.

About Our Cover:

When the latest in high technology, VLSI electronics, gets integrated with an older technology such as that found in automotive, the results are quantum leaps in safety, performance, and reliability. Like the wheel in our picture, the trend to automotive electronics is accelerating.

CUSTOMER SUPPORT

CUSTOMER SUPPORT

Customer Support is Intel's complete support service that provides Intel customers with hardware support, software support, customer training, and consulting services. For more information contact your local sales offices.

After a customer purchases any system hardware or software product, service and support become major factors in determining whether that product will continue to meet a customer's expectations. Such support requires an international support organization and a breadth of programs to meet a variety of customer needs. As you might expect, Intel's customer support is quite extensive. It includes factory repair services and worldwide field service offices providing hardware repair services, software support services, customer training classes, and consulting services.

HARDWARE SUPPORT SERVICES

Intel is committed to providing an international service support package through a wide variety of service offerings available from Intel Hardware Support.

SOFTWARE SUPPORT SERVICES

Intel's software support consists of two levels of contracts. Standard support includes TIPS (Technical Information Phone Service), updates and subscription service (product-specific troubleshooting guides and COMMENTS Magazine). Basic support includes updates and the subscription service. Contracts are sold in environments which represent product groupings (i.e., iRMX environment).

CONSULTING SERVICES

Intel provides field systems engineering services for any phase of your development or support effort. You can use our systems engineers in a variety of ways ranging from assistance in using a new product, developing an application, personalizing training, and customizing or tailoring an Intel product to providing technical and management consulting. Systems Engineers are well versed in technical areas such as microcommunications, real-time applications, embedded microcontrollers, and network services. You know your application needs; we know our products. Working together we can help you get a successful product to market in the least possible time.

CUSTOMER TRAINING

Intel offers a wide range of instructional programs covering various aspects of system design and implementation. In just three to ten days a limited number of individuals learn more in a single workshop than in weeks of self-study. For optimum convenience, workshops are scheduled regularly at Training Centers worldwide or we can take our workshops to you for on-site instruction. Covering a wide variety of topics, Intel's major course categories include: architecture and assembly language, programming and operating systems, bitbus and LAN applications.

CUSTOMER SUPPORT

CUSTOMER SUPPORT

Customer support is Intel's complete support service that provides Intel customers with hardware support, software support, customer training, and consulting services. For more information contact your local sales office.

After a customer purchases any system hardware or software product, service and support become major factors in determining whether that product will continue to meet a customer's expectations. Such support requires an international support organization and a breadth of programs to meet a variety of customer needs. As you might expect, Intel's customer support is quite extensive. It includes factory repair services and worldwide field service offices providing hardware repair services, software support services, customer training classes, and consulting services.

HARDWARE SUPPORT SERVICES

Intel is committed to providing an international service support package through a wide variety of service offerings available from Intel Hardware Support.

SOFTWARE SUPPORT SERVICES

Intel's software support consists of two levels of contracts. Standard support includes TIPS (Technical Information Phone Service), updates and subscription service (product-specific troubleshooting guides and COMMENTS Magazine). Basic support includes updates and the subscription service. Contracts are sold in environments which require product programs (i.e., IBM environment).

CONSULTING SERVICES

Intel provides field systems engineering services for any phase of your development or support effort. You can use our systems engineers in a variety of ways ranging from assistance in using a new product, developing an application, personalizing training, and customizing or tailoring an Intel product to providing technical and management consulting. Systems Engineers are well versed in technical areas such as microcommunications, real-time applications, embedded microcontrollers, and network services. You know your application needs, we know our products. Working together we can help you get a successful product to market in the least possible time.

CUSTOMER TRAINING

Intel offers a wide range of instructional programs covering various aspects of system design and implementation. In just three to ten days a limited number of individuals learn more in a single workshop than in weeks of self-study. For optimum convenience, workshops are scheduled regularly at Training Centers worldwide or we can take our workshops to you for on-site instruction. Covering a wide variety of topics, Intel's major course categories include: architecture and assembly language, programming and operating systems, buses and LAN applications.



When Intel invented the microprocessor in 1971, it created the era of microcomputers. Whether used as microcontrollers in automobiles or microwave ovens, or as personal computers or supercomputers, Intel's microprocessors have always offered leading-edge technology. In the second half of the 1980s, Intel architectures have held at least a 75% market share of microprocessors at 16 bits and above. Intel continues to strive for the highest standards in memory, microcomputer components, modules, and systems to give its customers the best possible competitive advantages.

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

Above, BITBUS, COMMputer, CREDIT, Data Pipeline, FASTPATH, GENIUS, i, i², ICE, iCEL, iCS, iDBP, iDIS, i²ICE, iLBX, i_m, iMDDX, iMMX, Inboard, Insite, Intel, intel, intelBOS, Intel Certified, Inteleview, intelligent Identifier, intelligent Programming, Inteltec, Intellink, iOSP, iPDS, iPSC, iRMK, iRMX, iSBC, iSBX, iSDM, iSXM, KEPROM, Library Manager, MAP-NET, MCS, Megachassis, MICROMAINFRAME, MULTIBUS, MULTICHANNEL, MULTIMODULE, MultiSERVER, ONCE, OpenNET, OTP, PC-BUBBLE, Plug-A-Bubble, PROMPT, Promware, QUEST, QueX, Quick-Pulse Programming, Ripplemode, RMX/80, RUPI, Seamless, SLD, SugarCube, SupportNET, UPI, and VLSiCEL, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix, 4-SITE.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

*MULTIBUS is a patented Intel bus.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Distribution
Mail Stop SC6-59
3065 Bowers Avenue
Santa Clara, CA 95051

Table of Contents

Alphanumeric Index	vii
Preface—VLSI Trends in Automotive Applications	viii
QUALITY AND RELIABILITY	
Chapter 1	
Automotive Divisions	1-1
Chapter 2	
Quality and Reliability Overview	2-1
Chapter 3	
Reliability	3-1
Chapter 4	
Customer Quality Support	4-1
MEMORY PRODUCTS	
Chapter 5	
Memory Technologies	5-1
Chapter 6	
DATA SHEETS	
27C64/87C64 64K (8K x 8) CHMOS Production and UV Erasable PROMs	6-1
27C128 128K (16K x 3) CHMOS Production and UV Erasable PROMs	6-15
27C256 256K (32K x 8) CHMOS Production and UV Erasable PROMs	6-26
87C257 256K (32K x 8) CHMOS UV Erasable PROM	6-39
68C257 256K (32K x 8) CHMOS UV Erasable PROM	6-50
CONTROL AREA NETWORK (CAN)	
Chapter 7	
IN-VEHICLE NETWORKING BACKGROUND INFORMATION	
In-Vehicle Networking—Serial Communication Requirements and Directions	7-1
A High Performance Solution for In-Vehicle Networking—Controller Area Network (CAN)	7-14
Chapter 8	
82526 Controller Area Network Chip Architectural Overview	8-1
MCS®-51 FAMILY MICROCONTROLLERS	
Chapter 9	
Architectural Overview of the MCS®-51 Family of Microcontrollers	9-1
Chapter 10	
Hardware Description of the 8051, 8052 and 80C51	10-1
Chapter 11	
Hardware Description of the 80C51FA/83C51FA/87C51FA	11-1
Chapter 12	
MCS®-51 Programmer's Guide and Instruction Set	12-1
Chapter 13	
87C51GB/83C51GB/80C51GB CHMOS Single-Chip 8-Bit Microcontroller Architectural Overview	13-1
Chapter 14	
DATA SHEETS	
8031/8051/8031AH/8051AH/8032AH/8052AH/8751H/8751H-8 8-Bit Control-Oriented Microcomputers	14-1
8051 AHP 8-Bit Control-Oriented Microcontroller with Protected ROM	14-16
8751BH Single-Chip 8-Bit Microcomputer with 4K Bytes of EPROM Program Memory	14-26
8052BH/8032BH Single-Chip 8-Bit Microcomputer	14-38

Table of Contents (Continued)

8752BH Single-Chip 8-Bit Microcomputer with 8K Bytes of EPROM Program Memory	14-47
80C31BH/80C51BH/87C51 CHMOS Single-Chip 8-Bit Microcontroller	14-59
80C51FA/83C51FA/87C51FA CHMOS Single-Chip 8-Bit Microcontroller	14-73
80C51GB/83C51GB/87C51GB CHMOS Single-Chip 8-Bit Microcontroller	14-89
APPLICATION NOTES	
AP-70 Using the Intel MCS®-51 Boolean Processing Capabilities	14-105
AP-252 Designing with the 80C51BH	14-150
AP-410 Enhanced Serial Port on the 83C51FA	14-174
ARTICLE REPRINT	
AR-517 Using the 8051 Microcontroller with Resonant Transducers	14-182
DEVELOPMENT SUPPORT TOOLS	
8051 Software Packages	14-187
ICETM-5100/252 In-Circuit Emulator for the MCS®-51 Family of Microcontrollers ..	14-195
MCS®-96 FAMILY OF MICROCONTROLLERS	
Chapter 15	
MCS®-96 Architectural Overview	15-1
Chapter 16	
MCS®-96 Instruction Set	16-1
Chapter 17	
MCS®-96 Hardware Design Information	17-1
Chapter 18	
80C196KA Advanced CHMOS Microcontroller Architectural Overview	18-1
Chapter 19	
DATA SHEETS	
809X-90/839X-90	19-1
809XBH/839XBH/879XBH Advanced 16-Bit Microcontroller with 8- or 16-Bit External Bus	19-30
APPLICATION NOTES	
AP-248 Using The 8096	19-47
AP-406 Analog Acquisition Primer	19-149
ARTICLE REPRINTS	
AR-321 High Performance Event Interface for a Microcomputer	19-173
AR-375 Motor Controllers Take The Single-Chip Route	19-178
DEVELOPMENT SUPPORT TOOLS	
MCS®-96 Software Development Packages	19-184
iSBE-96 Development Kit Single Board Emulator and Assembler for MCS®-96	19-194
VLSICETM-96 In-Circuit Emulator for the 8X9X Family of Microcontrollers	19-202
ICETM-196PC Real-Time Transparent 80C196 In-Circuit Emulator	19-212
ADDITIONAL INFORMATION	
Chapter 20	
APPLICATION NOTES	
AP-125 Designing Microcontroller Systems for Electrically Noisy Environments	20-1
AP-155 Oscillators for Microcontrollers	20-23

Alphanumeric Index

27C128 128K (16K x 8) CHMOS Production and UV Erasable PROMs	6-15
27C256 256K (32K X 8) CHMOS Production and UV Erasable PROMs	6-26
27C64/87C64 64K (8K x 8) CHMOS Production and UV Erasable PROMs	6-1
68C257 256K (32K x 8) CHMOS UV Erasable PROM	6-50
80C196KA Advanced CHMOS Microcontroller Architectural Overview	18-1
80C31BH/80C51BH/87C51 CHMOS Single-Chip 8-Bit Microcontroller	14-59
80C51FA/83C51FA/87C51FA CHMOS Single-Chip 8-Bit Microcontroller	14-73
80C51GB/83C51GB/87C51GB CHMOS Single-Chip 8-Bit Microcontroller	14-89
8031/8051/8031AH/8051AH/8032AH/8052AH/8751H/8751H-8 8-Bit Control-Oriented Microcomputers	14-1
8051 AHP 8-Bit Control-Oriented Microcontroller with Protected ROM	14-16
8051 Software Packages	14-187
8052BH/8032BH Single-Chip 8-Bit Microcomputer	14-38
809X-90/839X-90	19-1
809XBH/839XBH/879XBH Advanced 16-Bit Microcontroller with 8- or 16-Bit External Bus	19-30
82526 Controller Area Network Chip Architectural Overview	8-1
87C257 256K (32K x 8) CHMOS UV Erasable PROM	6-39
87C51GB/83C51GB/80C51GB CHMOS Single-Chip 8-Bit Microcontroller Architectural Overview	13-1
8751BH Single-Chip 8-Bit Microcomputer with 4K Bytes of EPROM Program Memory	14-26
8752BH Single-Chip 8-Bit Microcomputer with 8K Bytes of EPROM Program Memory	14-47
Architectural Overview of the MCS®-51 Family of Microcontrollers	9-1
AP-125 Designing Microcontroller Systems for Electrically Noisy Environments	20-1
AP-155 Oscillators for Microcontrollers	20-23
AP-248 Using The 8096	19-47
AP-252 Designing with the 80C51BH	14-150
AP-406 Analog Acquisition Primer	19-149
AP-410 Enhanced Serial Port on the 83C51FA	14-174
AP-70 Using the Intel MCS®-51 Boolean Processing Capabilities	14-105
AR-321 High Performance Event Interface for a Microcomputer	19-173
AR-375 Motor Controllers Take The Single-Chip Route	19-178
AR-517 Using the 8051 Microcontroller with Resonant Transducers	14-182
Hardware Description of the 8051, 8052 and 80C51	10-1
Hardware Description of the 80C51FA/83C51FA/87C51FA	11-1
iSBE-96 Development Kit Single Board Emulator and Assembler for MCS®-96	19-194
ICETM-5100/252 In-Circuit Emulator for the MCS®-51 Family of Microcontrollers	14-195
ICETM-196PC Real-Time Transparent 80C196 In-Circuit Emulator	19-212
MCS®-51 Programmer's Guide and Instruction Set	12-1
MCS®-96 Architectural Overview	15-1
MCS®-96 Hardware Design Information	17-1
MCS®-96 Instruction Set	16-1
MCS®-96 Software Development Packages	19-184
VLSICETM-96 In-Circuit Emulator for the 8X9X Family of Microcontrollers	19-202

VLSI TRENDS IN AUTOMOTIVE APPLICATIONS

In the late 1970s, concerns about environmental pollution, long lines at the gas pumps and federal standards for fuel economy created unprecedented challenges for the automotive industry. A primary source of new capability, the microcomputer, arrived on the scene and started the "Electronic Age" of the automotive industry. This was a time characterized by a shift away from independent components to increasingly sophisticated control systems which linked components together into an engine control system. It was also a time of Very Large Scale Integrated Circuits (VLSI). The first appearance of VLSI microcomputer-based engine controls occurred in 1976 with the MISAR Electronic Engine Control System on General Motor's (GM) Toronado. VLSI microcomputer-based engine controls from Chrysler and Ford followed quickly with Ford introducing its first generation of electronic engine controls, called EEC-I, in 1978. Microcontroller has all but supplanted the microcomputer terminology because of the control application and the advent of the microcomputer system.

As quickly as the capability of the microcomputer was proven for engine control, automotive engineers realized the tremendous potential of the microcomputer for other functions. The next few years witnessed a dramatic increase in the number of engine functions controlled. The VLSI industry continued to advance technology by downward scaling of the basic transistor size. As functions and memory usage increased, more logic and memory were integrated onto the microcomputer chip.

In the early 1980s as the application of the microcomputer for environmental and fuel economy control became better understood, the focus of the automakers shifted to performance. Cars now had to be "fun to drive" and give the driver a feeling of control. It was not sufficient just to meet federal regulations. New generations of engine controls and, more recently, electronically modified suspensions and anti-lock braking systems emerged.

By 1983, Ford had evolved to its fourth generation of engine controls, or EEC-IV. EEC-IV is designed around an Intel custom 16-bit microcomputer chip set, comprised of the 8061 CPU and the companion 8361 memory chip. As controlled engine functions continued to increase, and as automotive engineers became more familiar with the power and flexibility of the microcomputer, memory requirements increased dramatically. In 1978, for example, the Ford EEC-I used only 1542 (1.5K) bytes of program memory. By 1983 EEC-IV used 16 Kbytes, more than a 10X increase in six years. The memory technology used during the early days of microcontrolled engines was Read Only Memory (ROM). ROM technology implies that the engine control program was "hardwired" into logic in the ROM

device and could not be altered. In addition, ROM technology is usually characterized by a typically long manufacturing cycle of eight to ten weeks or longer.

Occurring simultaneously with the rapid increase in program memory was, the increasing number of unique engine software programs. Unique "codes" included those for 4-, 6- and 8-cylinder engines; California vehicles; those with and without automatic transmissions; and even some unique codes for high-altitude locations. The automotive software engineer soon found himself to be the bottleneck to "Job #1". Twenty-five or more codes were not unusual, and automakers were forced to stagger the final sign-off of each code to prevent an overload situation for the software engineer and the VLSI suppliers who were faced with this on-rush of new codes as "Job #1" approached. Automakers were also faced with monumental material control logistics during code start-up and change-over. Last minute changes in codes could and sometimes did cause vehicle line stoppages. It was time for a change.

The solution to the problems resided with the Erasable Programmable ROM (EPROM). EPROMs allowed codes to be altered electronically at the engine system manufacturing plant, thus giving the automaker the time to finalize codes and to better control inventories. In 1982, GM was one of the first to use EPROM technology for engine control via Intel's 2716, a 2-Kbyte device. By 1984, Ford was using an Intel-designed custom 16-Kbyte EPROM in EEC-IV. Today EEC-IV uses an Intel 32-Kbyte custom EPROM, the 8763. EPROM technology played an important role in this phase of engine controls and will continue to do so in the future as transistor scaling and memory densification continue.

As mentioned earlier, the focus has now shifted to performance and driver control. This has brought on the introduction of microcomputer-controlled anti-lock braking systems (ABS) and electronically-controlled ride and semi-active suspension systems. An example of a popular ABS is the Alfred Teves GMBH system. Introduced in 1985 on luxury cars, but currently in use on many popular U.S. and European sedans, the Teves ABS is based on Intel's MCS-8051 family of microcomputers. The system controls the brake fluid pressure to prevent the wheels from "locking up" (and the car from losing steering control) in severe braking conditions.

Robert Bosch GMBH and Kelsey Hayes in the U.S. have also emerged as leaders in ABS. The Kelsey system was introduced this past year on various U.S.-manufactured light trucks. It controls only the rear brakes of the truck. Since rear axle loading varies significantly,

rear braking control keeps total truck braking in balance.

Bosch has taken ABS to new levels of sophistication by integrating traction control with very little additional hardware or software. Traction control prevents the spin of the driven wheels, thus maintaining stability if vehicle acceleration is excessive. The new Bosch system was introduced this year on European luxury sedans.

A number of different ride control systems have been introduced over the past few years, primarily on luxury or sports vehicles. These systems use microcomputers to adjust shock absorber damping to provide the driver with the choice of a "firm" or "soft" ride. An example of this kind of system can be found as an option on the 1987 Ford Thunderbird. Soon to be released systems, described as semi-active, move a step closer to total control of the spring rates as well as shock absorber damping.

Where then does the "Electronic Age" lead to in the future? Engine controls will evolve into Powertrain Control Computer Systems (PCCS). New levels of sophistication will be achieved in modeling the combustion process and in the ability to control the combustion of each cylinder for peak performance and fuel economy, while reducing pollutants even further. Electronic automatic transmission controls will be integrated into the PCCS. Cruise control, already integrated into some engine controls, will evolve to "drive-by-wire" to become a component of the PCCS.

The VLSI industry will also move ahead, continuing to produce more powerful and capable microcomputers. Sub-micron technology now on the horizon is capable of packing more than a million transistors of random logic onto a cost-effective device. Processing power, calculated in terms of millions of instructions per second (MIPS), will increase at least an order of magnitude by the early 1990s into the range of 20 to 30 MIPS. The transition to 16-bit and 32-bit microcomputer systems with more sophisticated input/output circuitry (I/O) will emerge. Memory growth will also continue to grow, reaching 1- to 2-Megabytes. In-circuit alterable memories will become the norm. These memories will allow the automaker to evolve to in-module programming of the codes for the PCCS at the end of the assembly line. Both EPROM and the newly emerging Electrically Erasable PROM (EEPROM) technology will facilitate this movement.

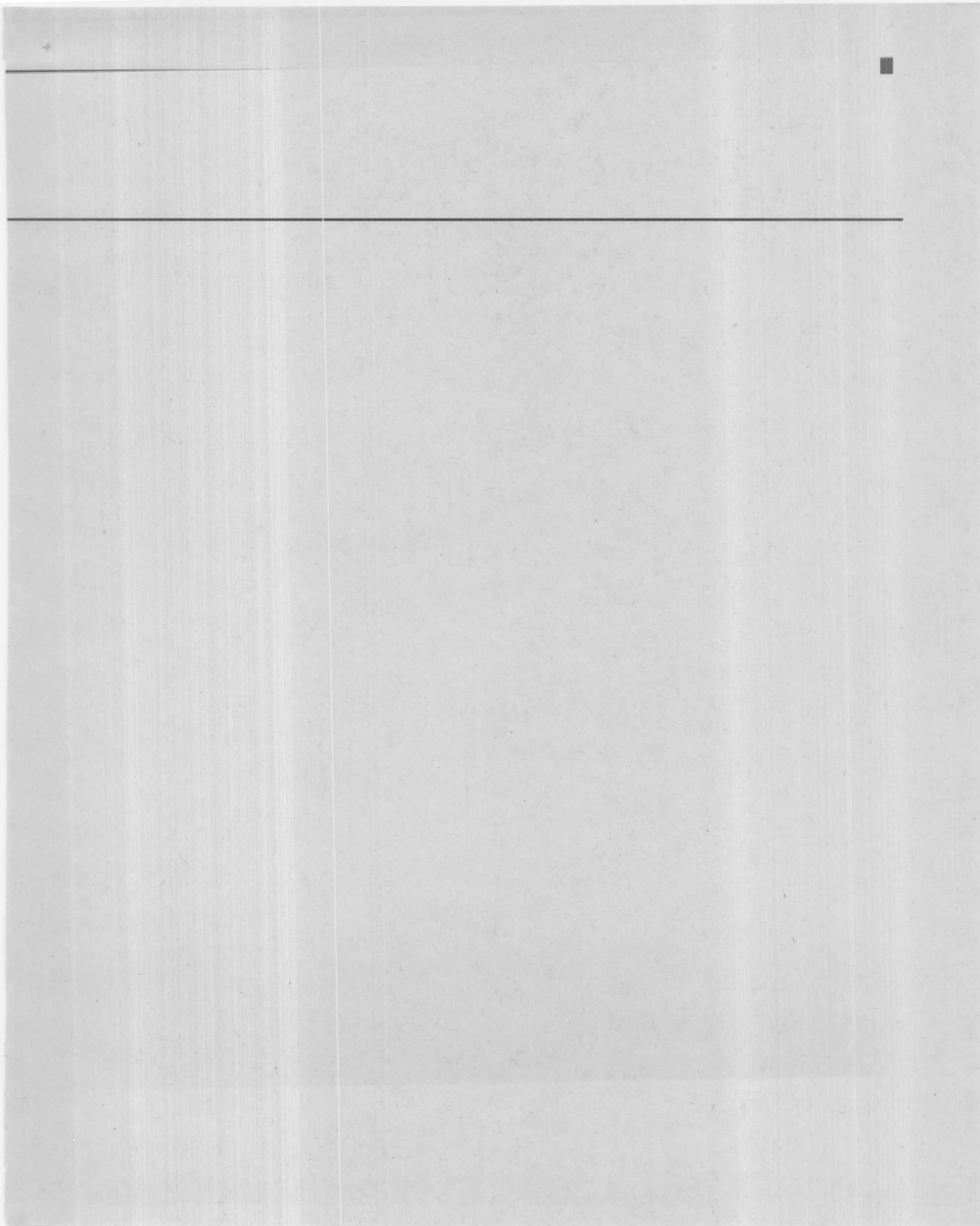
What about systems other than PCCS? A University of Michigan study in 1986 forecasts a 50 percent usage of ABS and 40 percent usage of ride and active suspension systems by 1995 for U.S. vehicles. An approach to facilitating this growth parallels that of engine controls in

the form of a Vehicle Control Computer System (VCCS). Integration of ABS, traction, suspension, and perhaps steering will form the VCCS. Likewise, a similar integration will occur for passenger compartment controls such as the Instrument Cluster, Climate Control, Trip Computer, and other comfort and convenience controls. Ford has referred to this as Stage III of the "Electronic Age".

What then becomes the next technological barrier to hurdle? An estimate by Chrysler predicts 250 new vehicle electrical circuits will be added between 1985 and 1989. Since 250 new circuits were added from 1981 to 1985, some 500 new circuits will be added to the average car in eight years. Integration of new electronic systems will continue this growth. The solution to circuit complexity seems to reside in multiplexing. Multiplexing uses the concept from office automation of linking systems together on a common path or bus to share information. For the automakers, both data and control information must be conveyed reliably in a very electrically "noisy" environment. Many automakers are now involved in defining the requirements of multiplexing in the future. The Society of Automotive Engineers and the American Trucking Association in the U.S. as well as the International Standards Organization, have formed subcommittees to focus on understanding multiplexing and to set standards. This task is made more difficult by the varying needs from simple on/off control to very high-speed sharing of data and control information for complicated electronic systems such as the PCCS.

Robert Bosch GMBH and Intel have collaborated on one such multiplexing scheme referred to as Controller Area Network (CAN). Designed for a broad range of applications where data and control information are important, CAN utilizes technology to allow the automaker to easily integrate communications into existing electronic control systems. By the mid 1990s, with the advent of cost-effective silicon switches, multiplexing schemes will become commonplace.

In conclusion, VLSI microcomputer technology ushered in the "Electronic Age" for the automotive industry and will continue to sustain it for tomorrow. As more complex microcomputer-controlled systems are required over the next 10 to 15 years, continued downward scaling of the basic transistor will result in higher performance and higher levels of integration of both the microcomputer and memory devices. Scaling, along with unique memory technologies such as EPROM and EEPROM will offer new levels of flexibility, adaptability, and diagnostics to the automakers. The technology of multiplexing will bring the "Electronic Age" of vehicle electronics to Stage III with sophisticated, yet cost-effective electronic computer control systems.



Automotive Divisions

1

CHAPTER 1 AUTOMOTIVE DIVISIONS

Intel Components consists of a number of product operations, which historically have started out in Santa Clara, CA, as smaller segments of a larger product division, and were reorganized as separate entities when their market segments became sufficiently large. Ultimately, the groups moved to other locations as they grew in product breadth and volume. Consequently, although Intel utilizes three major silicon technologies and one major non-silicon technology, Intel Components is now composed of many product operations. The three silicon technologies, random access memories (RAMs), non-volatile erasable programmable read only memories (EPROMs and E²PROMs), and microcomputer chips account for nine operations.

Rather than attempt to describe each operation separately, this section is divided into major product areas, some of which are not specifically identified with any product division. For example, military products are specially processed and tested products, otherwise identical to Intel standard products manufactured by other product divisions. Microprocessors and microcontrollers utilize the same fab processes; the divisions reflect more customer orientation than internal distinctions.

In order to ensure product quality and reliability consistency between divisions, Intel Components Q & R is represented in each division by a distinct Q & R staff, all reporting to the Director of Components Q & R.

In addition, the various product divisions are linked by a Manufacturing Managers Council (MMC). Since each division in essence purchases its products from Intel's Wafer Fab, Assembly and Test Manufacturing Divisions, it is essential that division-to-division differences be minimized and common resources efficiently utilized; this is the charter of the MMC.

The functions of the product divisions are described in terms of memories, non-volatile memories, and microprocessors, with three special sections on analog (telephony) products, automotive products and on military products. For more detailed descriptions, the reader is referred to the appropriate product division handbook or product catalog.

AUTOMOTIVE PRODUCTS

Introduction

The Automotive group is involved in designing, manufacturing, and marketing microcontrollers and memories for the automotive world. These integrated circuits are primarily used for engine control, dashboard control and display, suspension systems, braking, and power steering systems. Automotive network systems will soon eliminate the need for mechanical linkages between the operator and the automobile's functions. The Automotive integrated circuits perform both analog and digital processing and control in extremely harsh environments that demand the highest levels of quality and reliability.

The expected life of integrated circuits in automotive applications is in the range of ten to twenty years. Special considerations must be taken in design, processes and materials to meet these requirements. Testing of these highly integrated analog and digital circuits usually extends beyond those of normal commercial products. Both the analog and high frequency applications require specialized test equipment of great accuracy and repeatability. Due to the numerous methods of testing microcontrollers, customer correlation activities take on even more significance. Because of these unique requirements, Intel has developed specialists in automotive applications, quality, reliability, and design.

Quality and Reliability Organization

The Automotive Quality and Reliability Organization is well versed in both quality and reliability procedures for microcontrollers and memories in automotive applications. The quality and reliability of the product are integrated into the design and carried into the wafer fabrication assembly and test areas. The product is then subjected to qualification procedures that meet the high standards of the automotive industry.

The Automotive Q & R organization shares common wafer fabrication, assembly and test information which ensure the latest technology improvements and methods are used in the Automotive products. After the initial design, Q & R participates in three programs to ensure customer satisfaction. These three areas are: Reliability Testing, Monitors, and Product Quality Improvement.

Reliability Testing

The Automotive Q & R organization qualifies the product to ensure that both the customer and Intel specifications are met or exceeded. This reliability testing ensures conformance to the actual process. These reliability tests include some of the following:

1. High Temperature Dynamics Burn-In/Life
2. High Temperature Storage
3. Package/Die Stress
4. Electrostatic Discharge Characterization
5. Moisture Resistance

Monitors

Once a technology/product is qualified, production monitors are established to ensure that reliability objectives are met on a continuing basis. Samples from standard production material representing each technology, product family, and manufacturing plant are subjected to reliability testing. Analysis allows for rapid identification of potential problems and subsequent corrective action. If at any time a failure rate exceeds predetermined limits, production shipments are held until corrective action has taken place.

Product Quality Improvement

Quality/Reliability improvements start during the initial phases of a product's development and last throughout its life cycle. As the product/technology matures, lower defect levels are obtained. Using data and failure analysis from qualification and monitor activities, dominant mechanisms are identified. These mechanisms are fully characterized, resulting in optimized or new screens and/or process control improvements. Each new generation builds on this knowledge. While Intel employs the use of screening techniques to improve quality and reliability in the early stages of a product's life cycle, the longer term goal is to achieve reduced product failure rates through process and/or process control improvements.

Intel also uses the failure analysis/correlation request (FA/CR) system to achieve mature product quality/reliability improvements. Through the FA/CR program any customer can request Intel engineers to investigate potential problems. Analysis is performed on the returned devices, and a detailed report is written and sent to the customer explaining the findings, cause, and actions taken to correct the failure.

A questionnaire is also sent to the customer with a stamped/addressed envelope for comments on the FA/CR system.

Automotive customer confidence is essential to achieve minimal or no incoming inspection. A formal program that regularly feeds back to Intel information on defects, returned material, and other data can effectively eliminate any problems and increase awareness of product quality and reliability.

AUTOMOTIVE MEMORY PRODUCTS

Introduction

The Intel operation responsible for Automotive Memory Devices is located in Folsom, California. This operation is supported by two product-specific groups within Q & R: a Product Assurance group which addresses outgoing quality issues, and a Quality/Reliability Engineering group which ensures that improvement programs constantly maintain Intel's leadership position in supplying high-reliability products.

A prime responsibility of Product Assurance is to perform the final product inspections designed to ensure that product quality goals are continually met. Product-specific quality data from Intel's centralized test facilities are also forwarded to and monitored by Product Assurance. The data are routinely trended for the three major categories of defect modes, electrical, hermeticity, and mechanical/visual. A detailed breakdown of the failure modes is regularly reviewed to assure control and to assure that timely corrective actions are effective in eliminating defects from any source including design.

Product Assurance is also responsible for administering and/or implementing several of the customer-related quality programs within Intel. An example of one of the programs is the Qualification Detail Form (QDF) System, which provides customers with evaluation units built to negotiated specifications. A second example is the Failure Analysis/Correlation Request (FA/CR) System, which provides customers with an avenue to obtain a thorough analysis by Intel. Intel provides the customer with preliminary verbal reports and a written EER (Engineering Evaluation Report) which defines the cause. It also defines the required corrective action taken by Intel and recommends action applicable to the customer. A third example is the Customer Specification System, which ensures that negotiated product flows, tests, or monitors are in place and operative at all times, according to customer requirements.

Product Quality/Reliability Engineering

The primary responsibility of Product Q & R is to ensure that Intel products meet the corporate goals for quality and reliability at all times during the product life cycle. This responsibility is shared with other reliability engineering groups within Intel when appropriate. In Oregon, for example, a Technology RE group works closely with Technology Development during the development cycle of a new RAM technology. As the development cycle draws to a close, the Product Q & R group involvement increases to ensure that manufacturing controls are developed and implemented. A coordinated effort ensures that the qualification exercise addresses all known or potential hazards. New types of packages and/or new assembly processes are likewise qualified by a coordinated effort with Intel's Package Q & R group in Chandler, Arizona.

The qualification of a new product at Intel requires a large and meticulous effort, but the process does not stop at the initial qualification. Each new design or design iteration undergoes a separate qualification exercise, even though the manufacturing processes remain essentially unchanged. Particular attention is paid to layout-sensitive parameters that may impact quality or reliability, such as electrostatic discharge (ESD) sensitivity, AC marginality, and soft error rate. Product Q & R works closely with Design Engineering to ensure that quality and reliability are designed and built into these products.

EPROMS

Introduction

Small and inexpensive microprocessors have made possible scores of products ranging from simple toys to complex satellites. At the same time, microprocessors have created a fast-growing market for memories, the devices that store the programs and data that instruct the microprocessor.

The devices best suited for these applications are non-volatile memories that do not lose programmed data when power fails or is shut off. These memory devices must also allow data to be read quickly and do not require data to be changed during operation. Some allow programs to be separately altered when not in operation.

Non-volatile memories produced by Intel include Read Only Memory (ROM), Programmable ROMs (PROMs), Erasable PROMs (EPROMs), Electrically Erasable PROMs (E²PROMs) and Erasable Programmable Logic Devices (EPLDs). What distinguishes these various devices is the degree to which their contents can be changed.

Memory Types

ROMs

ROMs are preprogrammed by the semiconductor vendor to the customer's specifications. They are the least flexible non-volatile memory because they can be programmed only during wafer fabrication.

PROMs

As with ROMs, PROMs can be programmed only once—but by the customer. These devices allow the customer to inventory blank PROMs and program them when needed. EPROMs in plastic packages fit this category because they can only be programmed once; however, they have the advantage of being fully tested for speed, programmability, and data retention. Surveys of EPROM usage show that 80 percent of all EPROMs are programmed only once.

EPROMs

Erasable PROMs are more flexible than ROMs or PROMs because they can be programmed, erased, and reprogrammed by an OEM (original equipment manufacturer) or end user. Reprogramming an EPROM requires that it be erased by exposure to intense ultraviolet light for about 15 minutes through a UV transparent window in its package, and then electrically reprogrammed with new instructions. EPROMs now offer the same density and performance as ROMs. OTP EPROMs (one time programmable) are now also available in plastic windowless packages and are intended for the high volume automated end user or as ROM replacements.

EPLDs

Erasable Programmable Logic Devices are a new ASIC device. They currently use EPROM technology and are available in several architectures. They are intended for uses similar to logic arrays, PLDs, and a variety of other uses.

NVRAMs

Non-volatile RAMs have the characteristics of both E²PROMs and static RAMs. They read/write at microprocessor speeds like a static RAM. Non-volatile RAMs have an E²PROM memory cell for each bit of static RAM memory that stores non-volatile data. The non-volatile data can be stored in 10 ms and recalled in 10 μ s. NVRAMs are useful for power-fail applications where critical data must be quickly stored.

Technology

EPROMs and E²PROMs use a dual-layer polysilicon gate technology to permanently store charge. Floating gate technology is used to store a charge that makes the memory cell conducting or non-conducting, thus implementing a logical "1" or "0". The floating polysilicon gate is surrounded by high quality, thermally grown silicon dioxide, which gives these memory devices their excellent data retention capabilities.

EPROM Technology

The floating gate EPROM cell layout is shown in Figure 1a and in cross-section in Figure 1b. The memory cell utilizes a self-aligned floating gate to increase performance and density. The EPROM cell is initially in the neutral state with no charge on the floating gate, and the cell exhibits a threshold for conduction like a normal MOS transistor. The cell will conduct current when the access gate is selected. The memory cell is programmed via hot electron injection from the drain depletion region. A programmed cell has enough electrons on the floating gate so that the transistor does not conduct current, and the sense amp reads a programmed bit as a logical "0".

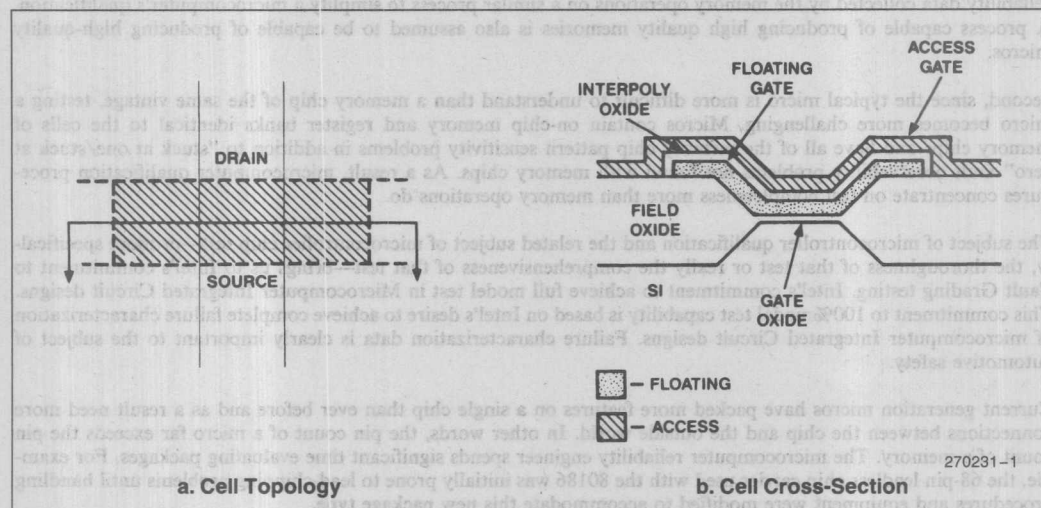


Figure 1. Floating Gate EPROM Cell

AUTOMOTIVE MICROCONTROLLERS AND PERIPHERALS

Introduction

Intel pioneered the first microprocessor, the 4004, in 1972. Since then, the market has grown dramatically, resulting in a wide range of device shapes, sizes, and capabilities. Applications are even more varied than the number of processors themselves. To support this growth, Intel has established three separate operations. The Microcontroller Operation is located in Chandler, Arizona, while the Microprocessor Operation is in Santa Clara, California, and the Peripheral Controller Operation is in Folsom, California.

Microcontrollers are intended for low-cost, high-volume, minimum chip count applications. Intel's Microcontroller Operation product lines include the MCS-48, MCS-51, and MCS-96. Microcontroller architectures minimize support circuitry required for operation; in many applications, the microcontroller is the only integrated circuit present. The operation also supports chips which extend the basic single-chip controller into board-level systems.

Micros vs. Memories

Testing micros is considerably different than testing traditional memory circuits. First, unlike RAMs and ROMs, a failing transistor is very difficult to locate in a typical microprocessor/microcontroller design. With memories, there are cost-effective techniques for mapping failing bits as seen by a tester to physical device locations. These techniques are very expensive for random logic and simply do not work on devices as complex as an 8051 or an 80286. In a micro, finding a bad transistor is literally like looking for a needle in a haystack. For this reason, it is difficult to perform fundamental reliability experiments using random logic designs. All the microcomputer operations use reliability data collected by the memory operations on a similar process to simplify a microcomputer's qualification. A process capable of producing high quality memories is also assumed to be capable of producing high-quality micros.

Second, since the typical micro is more difficult to understand than a memory chip of the same vintage, testing a micro becomes more challenging. Micros contain on-chip memory and register banks identical to the cells of memory chips and have all of the memory chip pattern sensitivity problems in addition to "stuck at one/stuck at zero" fault observability problems not found with memory chips. As a result, microcomputer qualification procedures concentrate on test completeness more than memory operations do.

The subject of microcontroller qualification and the related subject of microcontroller chip test—or more specifically, the thoroughness of that test or really the comprehensiveness of that test—brings us to Intel's commitment to Fault Grading testing. Intel's commitment to achieve full model test in Microcomputer Integrated Circuit designs. This commitment to 100% nodal test capability is based on Intel's desire to achieve complete failure characterization of microcomputer Integrated Circuit designs. Failure characterization data is clearly important to the subject of automotive safety.

Current generation micros have packed more features on a single chip than ever before and as a result need more connections between the chip and the outside world. In other words, the pin count of a micro far exceeds the pin count of a memory. The microcomputer reliability engineer spends significant time evaluating packages. For example, the 68-pin leadless chip carrier used with the 80186 was initially prone to lead-chipping problems until handling procedures and equipment were modified to accommodate this new package type.

Because they can easily find failing transistors, the memory operations perform the fundamental process reliability studies. In addition to training in electrical engineering, their reliability engineers also have strong backgrounds in solid-state physics. As a result, memory qualification procedures collect more lifetest device hours than the microprocessor operations do. On the other hand, the microcomputer reliability engineer is additionally concerned with testing and package issues not seen with memories. Fewer lifetest hours are collected and more time is spent on review of design, testing, and handling procedures.

The Q & R Interface

As the microprocessor market has grown, it has matured. The quality and reliability goals demanded of today's micros track the goals set for memories. New procedures and techniques are continuously developed to reach these goals. Procedures used by the microprocessor operations are based on procedures established by the memories. Intra-site forums and councils are used to communicate developments from other sites, and all operations at Intel use similar operating procedures. Differences in the product lines still cause the operations to vary in what they concentrate on, but common procedures and tools still allow all operations to benefit from the others' experience.

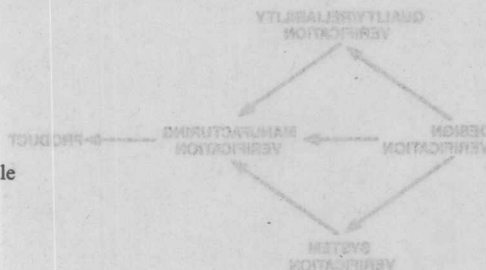
As mentioned, microcomputer operations depend on memory operations to establish initial process reliability and quality baselines. Product screening techniques developed by the memory operations, such as high voltage cell stress, are incorporated into microcomputer production flows. Microcomputer qualification procedures are then designed to detect non-conformance to goals, while the memory operations actually measure and quantify process/product performance. Reasonable effort is made to use memory product results and not reinvent wheels.

With micros, however, large dice and large packages raise additional concerns. Large random logic circuits contain less active gate area and more interconnection than do memories. Also, with high integration and high capability comes significantly more power consumption. To ensure no new hazards are created by these large dice, additional monitors and procedures are run within the microcomputer operations.

Reliability Testing

Microcomputer Integrated Circuit Reliability will be established utilizing the Automotive Integrated Circuit Qualification Plan. This Plan, mentioned earlier, is to be rooted in the SAE Integrated Circuit Qualification Plan which itself draws heavily on JEDEC 22A and MIL-STD-883 for methods. Some of the significant elements of the SAE Qualification Plan include:

- Temperature Cycling Molded Devices
- Thermal Shock
- Unbiased Autoclave
- Biased Humidity
- Operating Life
- Burn-In
- Power Temperature Cycle
- Internal Exams
- Electrostatic Discharge



Electrical Overstress (EOS)

This is a subject of particular importance to automotive engineers concerned with Automotive Integrated Circuits. Intel automotive, responding to this concern, places heavy emphasis on the EOS aspects of the microcomputer Integrated Circuit Design. ESD protection—a subset of EOS protection—is a highly visible aspect of the microcomputer Integrated Circuit Design and one that is judged with great care. The Intel Automotive Qualification Process is also very sensitive to the ESD issue.

Monitors

The Microcontroller Operation product line is not as broad as microprocessors or peripherals, but the products are manufactured at higher volume. This volume makes microcontrollers the primary monitor vehicles for 600 mil packages within the microcomputer operations. As in memory operations, a typical microcomputer monitor will subject 1000 units to a 48-hour 125°C dynamic burn-in, followed by a 1000-hour 125°C lifetest on 100 of the same units. Should the monitor process indicate failure results that exceed Intel's reliability goals, the Material Review Board (MRB) process will dictate the appropriate corrective action response.

Volume also makes it easier to measure and detect improvements in reliability and quality. New ideas in assembly, test, or finish are tried first on controllers and then, if favorable results are obtained, applied to the processor and peripheral lines. As an example, developments in wafer fabrication for high-reliability plastic were first applied to MCS-48 products and then applied to processors and peripherals.

Microprocessor and peripheral devices must work with many other chips in any given system, and they have interaction problems not found with single-chip controllers. The microprocessor and peripheral QRE operations together developed a procedure called Product Validation in which chips are evaluated in a system environment in addition to stand-alone performance against specification evaluations.

Product Validation

One of the quality improvement techniques mentioned earlier was Product Validation. Product Validation separates into four major activities where product reliability is only a fraction of the overall product quality program. Intel believes that a high quality microcomputer must:

1. Meet or exceed all specifications
2. Be produced without defects
3. Work in a system as intended
4. Work reliably

All new devices designed by microcomputer operations must pass all these verifications before volume shipments are allowed.

Product Validation is the methodology by which the microcomputer operations prove beyond reasonable doubt that a new product or a redesigned product meets all of its specifications and requirements and is production-worthy, as shown in Figure 2.

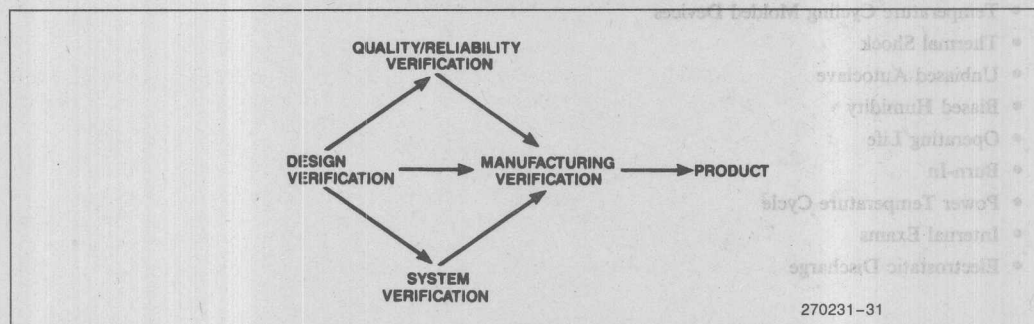


Figure 2. Product Validation Flow

Product Validation is a good example of how a Quality Reliability Engineer (QRE) works with the other departments at Intel. Within an operation, different departments are responsible for each of the four Product Validation activities.

Design Engineering is the group responsible for measuring the conformance of a new (or changed) product to all specifications. These measurements are made at multiple combinations of voltages between $V_{CC} \pm 15\%$, temperatures from -55°C to $+150^{\circ}\text{C}$, and at several operating frequencies. When measurements are completed, a design verification is submitted to the Product Validation committee for approval. No product is shipped for revenue until this report is approved.

Product Engineering is responsible for development of production test flows which produce defect-free product. Test results of material representing extremes of wafer fabrication variations are collected at test temperatures from -55°C to $+150^{\circ}\text{C}$. The product engineer then chairs a series of test review meetings which document test limits and sequences and provide a forum to discuss future test enhancements. The QRE audits corrective action plans generated at these test review meetings and any future changes to test flows.

Marketing coordinates a beta-site program with selected customers to demonstrate that Intel product works as documented in end-user systems. This customer feedback program has proven very effective in improving new product documentation and Intel's production test. This portion of the Product Validation procedure is considered important enough to cause Marketing to limit product shipments until system verification is complete.

The Quality/Reliability verification is the only portion of Product Validation actually performed by Quality and Reliability Engineering. In this portion of Product Validation, accelerated life stresses are applied to samples pulled from the actual manufacturing line and formal auditing of the total Product Validation procedure occurs.

In its simplest form, the auditing consists of reviewing the written plans submitted by Design Engineering, Product Engineering, Marketing, and Quality/Reliability Engineering against the requirements outlined in the Product Validation Procedure Specifications. Plans which do not meet requirements (i.e., by omitting an experimental procedure or failing to collect necessary data) are not approved. Approved plans, filed in the Q & R department, are a prerequisite to shipping product for revenue. After approval, the work called for in the plan is performed and required data is collected. These data are collected into a report, and this report is measured against its plan. The engineering group is finished when the report it submits documents that all points of the plan are completed. Revenue shipments are limited until all reports are approved.

New designs begin life with a required 48-hour burn-in imposed on 100% of the revenue product. The Q & R verification plan will also detail a burn-in evaluation which measures the product's infant mortality rate. The 100% burn-in is removed from the production flow only after the product has demonstrated conformance to Intel's infant mortality requirements. Additional screens are imposed if infant mortality rates are not met.

All fallout from any Q & R verification experiment or stress is analyzed to determine the failure mode. Since typical product qualifications produce very few failures, products on similar processes are combined into groups. These groups are reviewed for common problems. When a dominant failure mode can be identified, corrective action is taken to nip the problem in the bud. Intel does not wait for a product to fail qualification to initiate corrective action.

PRODUCT QUALITY IMPROVEMENT PROGRAMS

Product Validation was formally developed within Intel in 1980 and addresses quality improvement methodologies for new products and design changes in an older product. Quality improvement, however, never stops. After product validation, product goes into production, and for product developed before 1980, Intel has several other programs to improve product quality. The wafer fabrication and assembly areas have improvement programs which benefit all products. Additionally, the product operations have programs to find and correct product-specific quality problems.

While the Failure Analysis/Correlation Report (FA/CR) is not unique to microcomputer operations, this procedure spearheads our mature product quality improvement. Through the FA/CR program, any customer can request Intel engineers to investigate potential problems. The FA/CR is a natural extension of the System Verification beta-site.

Product Engineering analyzes product failing at Final QA and takes corrective action, such as calibrating test equipment more often, tightening test limits, or in some cases, renegotiating specifications.

In all operations, QRE audits testing of selected products through the formal monitor program. In this program, units are pulled from the end of the manufacturing line and retested before reliability stressing begins. Units which fail this retesting step are useful in identifying critical product performance parameters. Product Engineering then trends these critical parameters, and by working with wafer fabrication engineers, is thus able to focus on key wafer fabrication processes to improve the product.

Overview



CHAPTER 2 QUALITY/RELIABILITY OVERVIEW

THE INTEL QUALITY AND RELIABILITY PHILOSOPHY

Intel is committed to the highest possible standards of quality, reliability and customer satisfaction in its products. Intel's founders knew that achieving the goal of technological leadership in VLSI was not enough: quality and reliability assurance programs had to be as advanced as the technical competence of the product itself.

Intel's quality assurance procedures have not stopped with the traditional methods of testing and failure analysis. As VLSI technology has grown—largely as a result of Intel's leadership in original research and development of materials, process technologies, computer-aided design, applications engineering, and reliability physics—so too has the technology base of the quality program. This knowledge has been gained by the close involvement of the quality organization in the product life cycle, from validation of design rules through post-sale customer service.

Intel achieves the highest standards of product quality and reliability by:

- Making quality goals and achievements an integral part of every business operation.
- Maintaining an extensive product qualification program.
- Constantly striving to improve the quality and reliability process.

Since it is management's philosophy that quality is everyone's responsibility, quality goals and achievements are an integral part of every business operation. The annual planning process and the five-year strategic long-range plan (SLRP) both include quality objectives, which are reviewed annually by Intel's executive staff. In addition, quality programs and goals are included in quarterly objectives, and progress is reviewed regularly.

The basic philosophy of achieving quality excellence is "do it right the first time". To support this, quality training, courses on reliability physics, quality program reviews, and training in applied statistical methods are provided to the entire staff on Intel.

All Intel products must pass a rigorous qualification program before they are released to the marketplace. Intel insists on building in quality and reliability for every product from the very beginning of a technology and product development cycle. After a product is qualified, strict controls and monitors are applied in the manufacturing process to ensure its quality level. All processes are audited regularly to ensure that they meet specifications.

Intel constantly strives to improve its quality and reliability program by allocating significant research and development resources to reliability physics, quality engineering, and failure analysis. Intel will continue to expand industry knowledge in the critical areas that affect quality and reliability.

Understanding Intel's quality and reliability methodologies is critical to recognizing the added value that is built into each Intel product.

THE INTEL QUALITY PROGRAM—AN OVERVIEW

Introduction

From its founding in 1968, Intel recognized that to become the leading world supplier of VLSI technology required programs to ensure the highest quality and reliability standards. These standards are in place. Intel product is at the forefront of technology. Intel quality and reliability are unmatched. In this handbook, the quality and reliability programs established at Intel will be discussed both conceptually and factually. In addition, a description of Intel matrix management, product divisions, manufacturing flow, and various support efforts provides a broad framework from which to understand Intel quality and reliability programs. In this section, a general description of these programs is given; details follow in subsequent handbook sections.

Intel Facilities and Standardization

Intel design, wafer fabrication, assembly, and test facilities span the globe. There are design centers in Asia and the United States. Wafer fabrication facilities are located at six U.S. sites and in Israel. Assembly is performed in the U.S., the Philippines, and Malaysia. Manufacturing test facilities are located in the Philippines, Malaysia, Puerto Rico, and the U.S., while divisional test facilities operate at various U.S. sites and have extensions in Japan and England. In spite of this potpourri of sites, Intel guarantees that a customer will not receive products that have a "personality" based on its site of manufacture. The answer is a rigid system of standards imposed on every site, not just by means of product inspection criteria or process specification, but also through such vital functions as training and quality measurements, audits, and calibration.

For example, the quality system requirements are the same for each site, leading to standardization of process control via the same inspections, monitors, and data reporting and trending. The data are therefore available for a uniform measurement of quality at each site. To further assure this standardization, all site quality and reliability managers report directly to the Director of Quality and Reliability and secondarily to the site functions.

Measurement of Quality

There are two internal gauges used to measure quality. The DPM report (Defects Per Million; 1000 DPM = 0.1% defective units) summarizes defect levels taken from samples at key inspection points. The second is the ITR (Internal Trouble Report) rate or the LRR (Lot Reject Rate), which are lot rejection rates based on pass/fail of any sample acceptance criteria. To reiterate, since all tests and methods of sampling are standardized, a measurement of overall site quality performance and data on the quality of particular products and lots are universally available, irrespective of site, process, or product. Yearly long-range goals are set for both DPM and ITR, and progress towards these goals is reviewed every month.

Using the example of Intel's test sites, the DPM and ITR sample plans at each Final Quality Assurance (FQA) group on every lot of product is shown in Table 1.

Table 1. Sample Plans

Commercial Product (Hermetically Sealed and Molded Plastic)			
	Lot Size	Sample Size	Accept On
Electrical Test	<10001	125	0
Mechanical/Visual	<1200	76	0
	>1200	129	1
Hermeticity Test*	<1200	76	0
	>1200	129	1

*(for hermetically sealed package-only)

Military Product (Hermetically Sealed Packages Only)		
	Sample Size	Accept On
Electrical Test	125	0
Mechanical/Visual	76	0
Hermeticity Test	76	0

Although the content of the electrical test varies with the product, the sampling and testing methodologies are the same. They are controlled under one specification, #25-1406, for the entire components test world. To further demonstrate this standardization, Intel has one specification for mechanical/visual inspection criteria, #20-500 (Intel Workmanship Standards), and one each for fine and gross leak hermeticity tests, #20-044 and #20-004, respectively. The Intel specification system is discussed in more detail in a later section of this handbook. The training procedures for these tests are written and distributed by a single group for each specification.

The dramatic impact on DPM trend for an automotive device—as seen by the customer—as a direct result of Intel's ongoing improvements in quality system can be seen in Figure 1.

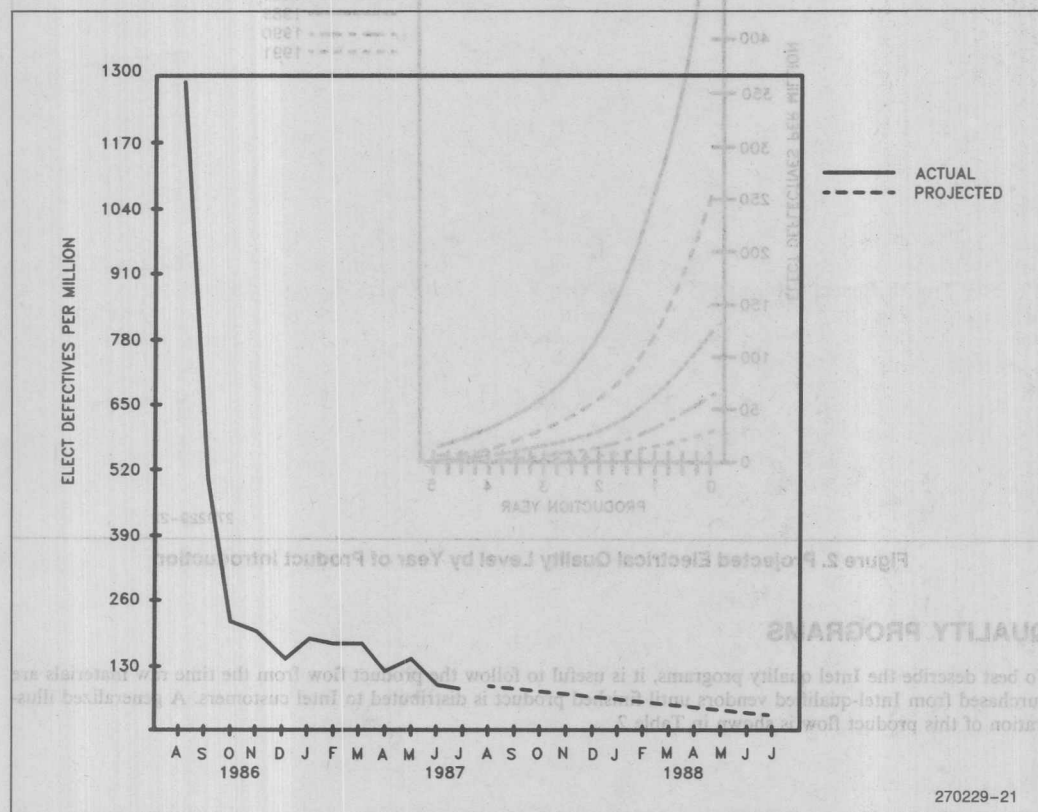


Figure 1. DPM for Intel Automotive Product

Continuing improvements in the quality systems can be seen in Figure 2. It is important to note that the Intel DPM target is set at zero.

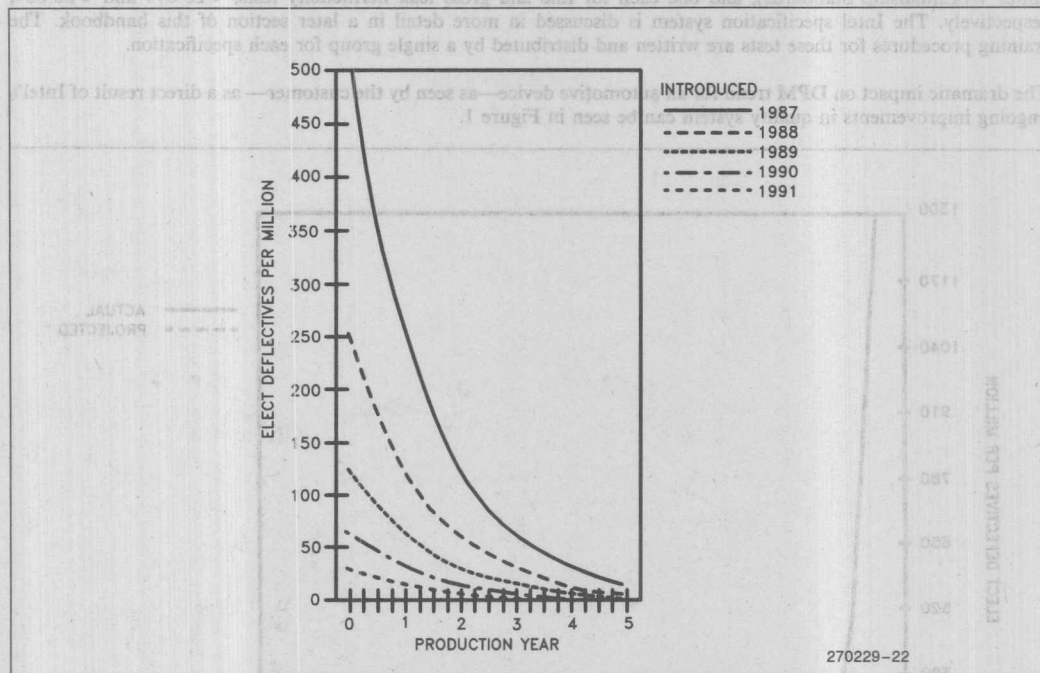


Figure 2. Projected Electrical Quality Level by Year of Product Introduction

QUALITY PROGRAMS

To best describe the Intel quality programs, it is useful to follow the product flow from the time raw materials are purchased from Intel-qualified vendors until finished product is distributed to Intel customers. A generalized illustration of this product flow is shown in Table 2.

Quality Function	Functional Flow	Reliability Function
Audits Vendor Quals	Vendor	Material Quals Material Monitors
Inspections Gates	Q & R	
Inspections Gates Process Controls	Fab	Process Monitors Plant/Process Quals
Inspections	Q & R	
Inspections Gates Process Controls	Assembly	Package Monitors Plant/Package Quals Audits
Outgoing QA	Q & R	
Inspections Gates Process Controls	Test	Product Monitors Product Quals Audits Reliability Monitors
Outgoing QA	Q & R	
Failure Analysis Correlation Reliability Analysis	Customer	Reliability Analysis

Incoming Materials

Intel manufactures no raw materials used in the production of integrated circuits. Silicon, chemicals, packages, and other materials are all purchased from Intel-qualified and -approved vendors. To ensure that only materials of the highest quality are used, Intel has an ambitious and far-reaching Vendor Quality Improvement Program (VQIP). This program operates under the following policies:

1. Purchase materials only from a small number of the highest-quality suppliers in the world.
2. Establish technical relationships with these few selected vendors and encourage strong technical interaction between vendor and Intel to mutually improve the entire technology.
3. Strongly relate material specifications to factory performance so as to ensure a strong linkage between the material specifications and factory needs.
4. It is Intel's intention to require compliance on the part of its vendors with the principles and tenets of SPC (statistical process control).
5. Improve product quality through Intel-vendor interaction to achieve low defect levels so that Intel incoming inspection becomes a data-gathering activity. This program is called C of C (Certificate of Conformance) in which vendors measure product quality to Intel specifications, and Intel accepts product based on vendor-supplied data.
6. Base purchase allocations on quantitative vendor comparisons where quality, delivery, total cost, and vendor responsiveness to Intel issues are considered in each vendor's ranking and rating.

The VQIP program has achieved the following goals:

1. Reduced a large vendor base to a select, world-class, highly qualified group of suppliers.
2. Reduced incoming defect rate of 10–15% to under 2000 DPM.
3. Reduced lot reject rate from 20% to under 2%.

4. Generated a system to purchase material based on a ranking system in which all qualified vendors have allocated to them a share of Intel's purchases based on monthly performance indicators.

Some of the vendor quality improvement data trends are shown in Table 3.

Table 3. Vendor Quality Trends

Commodity	1978		1980		1982		1984	
	DPM	LRR	DPM	LRR	DPM	LRR	DPM	LRR
Quartz		25%		20%		8%		3%
Silicon	25,000	18%	15,000	12%	6,000	8%	2,000	2%
Masks		30%		25%		20%		4%
Piece Parts	20,000	16%	15,000	12%	9,000	9%	2,000	3%

Wafer Fab

Wafer fab is the most sophisticated and complicated process in the production of VLSI devices. As wafer size and chip size increases and feature size decreases—because of chip complexity (see Figure 3) it becomes critical to reduce defect density levels. The yields if quality die are, to a very great extent, determined by wafer density levels. Since a single defect can render a large chip as inoperable as it can a small chip, the restrictions on defect level are continually being tightened. A given defect density on a wafer containing many small chips will be significantly less catastrophic than on the same wafer containing fewer large chips.

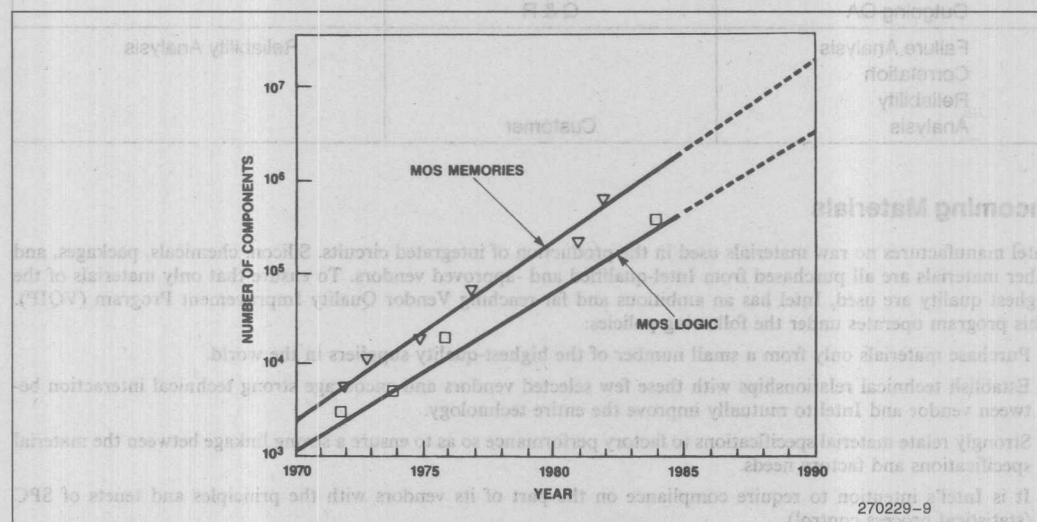


Figure 3. Chip Complexity Trends

The centerpiece program of the fab processing quality programs is SPC. This process whole heartedly supported by Intel ensures continuing stability of the fab manufacturing operation. This continuing stability may be the result of SPC-generated corrective actions or just the tangent result of SPC's constant presence. Additional quality programs for the fab operation includes:

- Continuous in-process controls and feedback to process steps
- Chip-level electrical testing
- Visual defect inspections
- Rationalization of final product test yields to process-related data

The overall result of such quality programs has been to raise die yields throughout the fab area to all-time highs. Because varying chip sizes are manufactured in each fab area using identical processes, the average yields are mathematically related to an artificial standard equivalent die size. The indicator of yield is then made relative to this standard and is called the Isodect yield. This is one of the standard parameters used to rate and rank the quality of Intel fabs, much as DPM and lot reject rate are used to rate and rank vendors and internal assembly and test performance. In Figure 4, there are three sets of wafer diameter curves, each for three different process maturity levels. As can be seen, new or immature processes, even using 150mm diameter wafers, produce less than ten 300 mil chips; similarly, average processes using 75mm wafers give few good chips. In order to obtain satisfactory yields in the few hundreds to a thousand chips per wafer, both high-quality processing and large diameter wafers are required. The improvement in fab Isodect yield is shown in Figure 5.

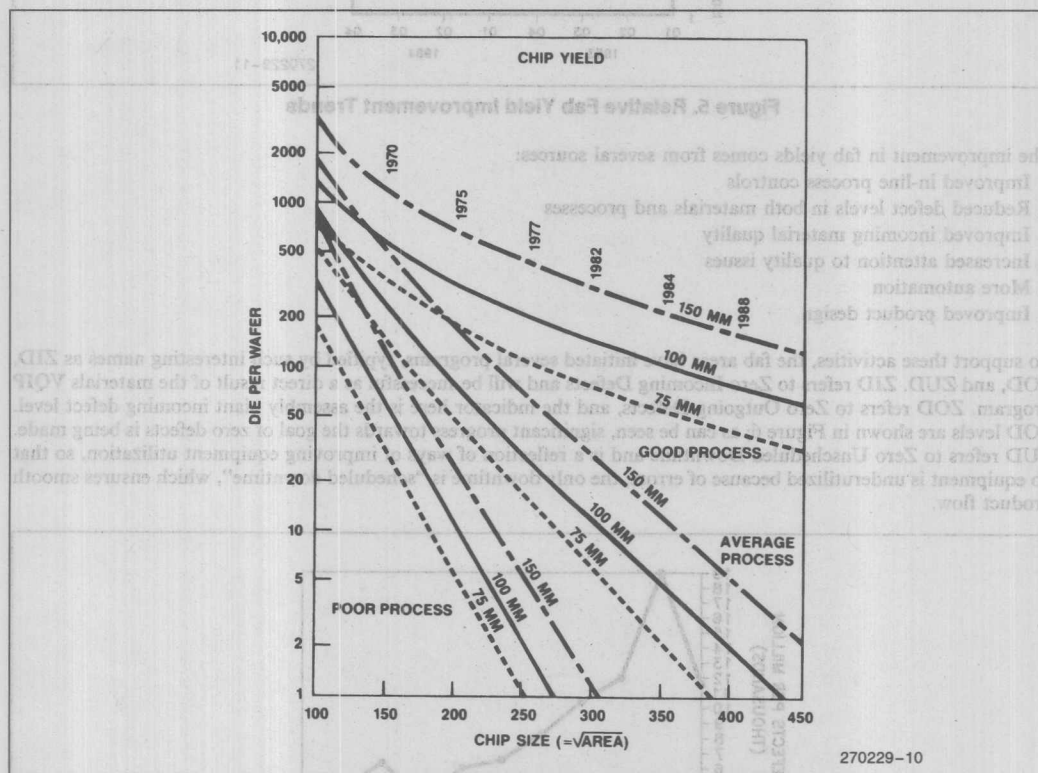


Figure 4. Relative Yield Vs. Chip Size for Different Process Maturities and for Different Wafer Diameters

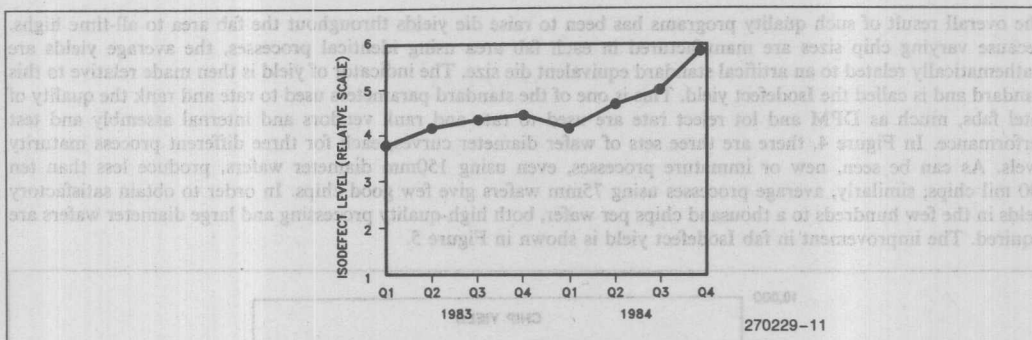


Figure 5. Relative Fab Yield Improvement Trends

The improvement in fab yields comes from several sources:

1. Improved in-line process controls
2. Reduced defect levels in both materials and processes
3. Improved incoming material quality
4. Increased attention to quality issues
5. More automation
6. Improved product design

To support these activities, the fab areas have initiated several programs, typified by such interesting names as ZID, ZOD, and ZUD. ZID refers to Zero Incoming Defects and will be successful as a direct result of the materials VQIP program. ZOD refers to Zero Outgoing Defects, and the indicator here is the assembly plant incoming defect level. ZOD levels are shown in Figure 6; as can be seen, significant progress towards the goal of zero defects is being made. ZUD refers to Zero Unscheduled Downtime and is a reflection of ways of improving equipment utilization, so that no equipment is underutilized because of errors; the only downtime is "scheduled downtime", which ensures smooth product flow.

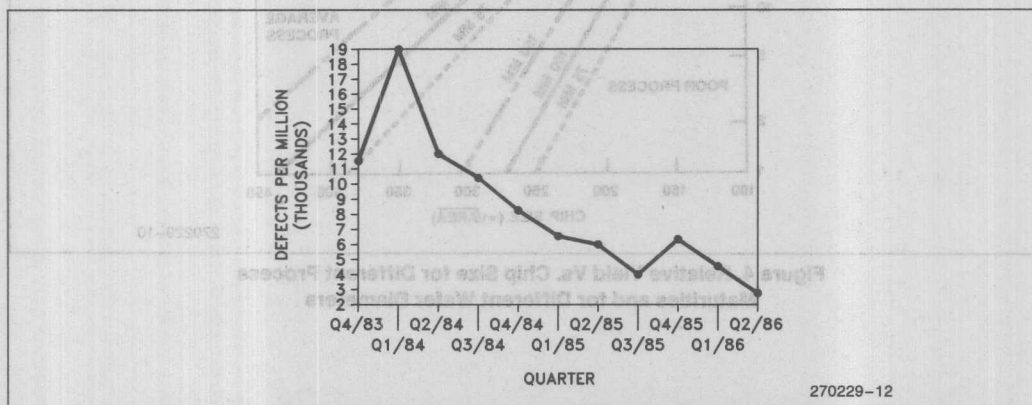


Figure 6. Visual ZOD Levels from Fab, or ZID Levels for Assembly

Statistical Process Control (SPC) underlies all these activities, and Intel has achieved notable success in implementing SPC in all manufacturing sites. Quality programs such as ZID, ZOD, ZUD, and SPC require extensive interaction with all the fab sites. Consequently, the fab world, even though centrally controlled, has evolved a series of internal councils to ensure program uniformity and standardization. The councils rank in hierarchy from the Fab

Managers Council (FMC), to Die Production Technical Review Board (DPTRB) and Die Production Administrative Review Board (DPARB), to Process Area Groups (PAGs). Through this extensive system of councils, whose meetings are held at each fab site on a rotating basis, quality, technology, and administrative standardization are maintained throughout all fabs.

With such high product complexity and with such great dependency on material and process parameters, fab yields would be non-existent without process and defect control. Inspection methods are costly, and features are often too small to be observed visually. Complexity is great and even electrical inspection is expensive and time-consuming. Hence, there is no alternative but to make product right. The slogan "zero defects" has been converted into an achievable goal. Fab and die production are discussed in the Wafer Fabrication section of this handbook.

Assembly

Each assembly process is less complex and sophisticated than its fab counterpart, but it must contend with significantly more individual units. Assembly handles three generic package types, with 8 to 15 package types within each package category. In addition, the assembly plants use an average of five assembly package parts for each chip, so that the total number of individual assembly steps is very large. Add to this the cost of product yield loss due to the package cost, and it is apparent that assembly quality is at least as important as fab quality.

Consequently, in parallel with Fab, Assembly has its own ZID, ZOD, and ZUD programs. Outgoing ZUD for Fab, and VQIP for Materials becomes ZID for Assembly, and ZOD for Assembly becomes ZID for Test. Some ZOD results for the last few years in assembly are shown in Figure 7; the Assembly ZID were shown in Figure 6. It is apparent that the three quality measures of Assembly—internal and external visual defects and hermeticity defects—have been steadily declining.

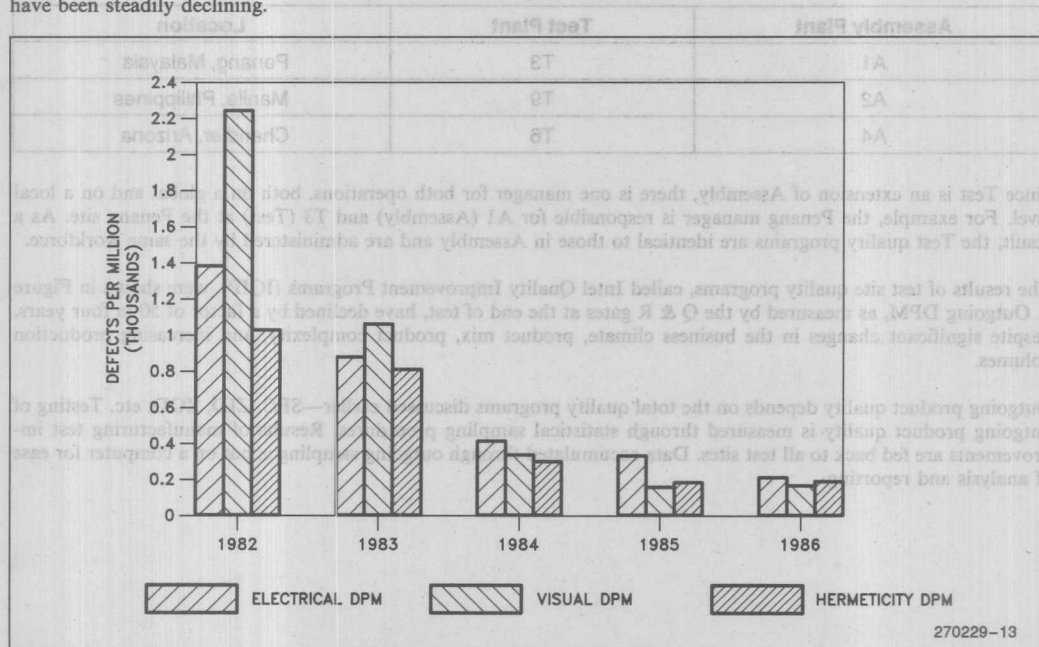


Figure 7. Assembly Defect Level at Final QA Gate

The Assembly operations are organized slightly differently from their Fab counterparts in that each assembly plant has its own Q & R function that reports directly to the Assembly/Test Q & R Manager located in Chandler, Arizona. Consequently, the council systems used to standardize fab quality programs are not required. Instead, tactical issues are handled by the unified functional operation, and councils concentrate on evaluation of strategic objectives.

Just as the dominant quality program in the fab world is SPC, so is SPC the dominant quality program in the assembly world. This process operating in conjunction with product lot inspection gates oversees the wide spectrum of Intel assembled product. Even here, however, as quality and consequently yield improves, the trend is to less inspection, and the goal is zero defects. In addition to the Intel-owned assembly facilities, assembly subcontractors are used. In order to maintain the high level of assembly quality irrespective of site of manufacture, the Q & R department has locally hired quality and reliability experts who operate within each subcontractor factory. The same set of specifications and requirements is used as in Intel-owned assembly, assuring shipment of uniform product.

The demands on quality of assembled product have increased. A single defect in the assembly of 125-pin PGA (Pin Grid Array) package will kill the device as surely as it would kill a 16-lead device. Defects are simply unacceptable. Because of the similarity in problems between Fab and Assembly—despite the vast differences in technology—Fab and Assembly representatives sit on the same manufacturing councils (hence the identical ZID, ZOD, ZUD programs). The commitment is to zero defects throughout both groups.

Test

Approximately 95 percent of Intel's product volume is tested at Intel's two manufacturing test sites. The remaining 5 percent of the product is tested in divisionally owned test sites. Therefore, the discussion of test quality will be restricted to the manufacturing sites, since in concept and in practice they determine outgoing quality levels.

Manufacturing test site location has been selected to be in close proximity to assembly plant location, and indeed, it is Intel's aim to send assembly plant output to the neighboring test site. Table 4 shows Assembly/Test proximity.

Table 4. Assembly/Test Proximity

Assembly Plant	Test Plant	Location
A1	T3	Penang, Malaysia
A2	T9	Manila, Philippines
A4	T6	Chandler, Arizona

Since Test is an extension of Assembly, there is one manager for both operations, both on a global and on a local level. For example, the Penang manager is responsible for A1 (Assembly) and T3 (Test) at the Penang site. As a result, the Test quality programs are identical to those in Assembly and are administered by the same workforce.

The results of test site quality programs, called Intel Quality Improvement Programs (IQIP), were shown in Figure 1. Outgoing DPM, as measured by the Q & R gates at the end of test, have declined by a factor of 50 in four years, despite significant changes in the business climate, product mix, product complexity, and increasing production volumes.

Outgoing product quality depends on the total quality programs discussed earlier—SPC, ZID, ZOD, etc. Testing of outgoing product quality is measured through statistical sampling procedures. Results of manufacturing test improvements are fed back to all test sites. Data accumulated through outgoing sampling is put on a computer for ease of analysis and reporting.

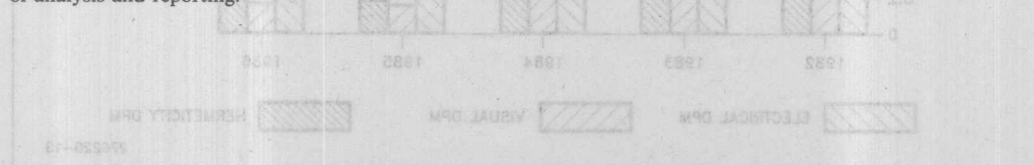


Figure 1. Assembly Defect Level at Final QA Gate

The Assembly operations are organized slightly differently from their Fab counterparts in that each assembly plant has its own Q & R function that reports directly to the Assembly Test Q & R Manager located in Chandler, Arizona. Consequently, the control systems used to standardize the quality programs are not replicated. Instead, functional issues are handled by the unified functional operation and council consisting of an evaluation of strategic objectives.

Product shipments to Intel Automotive customers are, of course, of paramount importance to Intel Automotive. However, prior to the onset of production shipments of new devices it is Intel Automotive's intention to supply to its customers descriptions of its quality plans on a per product basis. The document that will summarize the quality planning for each device will be known as a Control Plan. This document will be the basis for mutual agreement between Intel automotive and its customer relative to quality issues. The document will specify:

- Key device characteristics specified by the customer
- Key device characteristics specified by Intel Automotive

Customer Services

Because Intel Automotive serves a wide customer base with a variety of products and wants these products to be uniformly excellent, Intel Automotive has established a Customer Quality Engineering function. This function is responsible, through the various council systems, for maintaining standard and uniform quality and reliability goals for customers. Standard procedures for measuring DPM, for assessing Return Material Authorization rates, and for determining customer perception of Intel quality are thus ensured. Under this system, a customer need only interact with one quality group rather than communicate with a wide variety of quality engineers all operating under different guidelines. Failure analysis correlation, returned material analysis, customer and Intel issues can all be negotiated through this group.

It is Intel Automotive's intention to develop "relationships" with customers. A "relationship" with a customer is a special extra effort put forth by Intel to optimize the Intel customer interface. A "relationship" will provide mutual benefit to both Intel and the customer in the areas of: problem solving; future products; technology trends. The "relationship" approach lends itself to team-oriented problem solving disciplines.

Document Control and Technical Information Centers

There are several types of controlled documentation within Intel, in addition to library functions and technical memo distribution. Corporate and local document control centers maintain two levels of specifications, Intel Policies and Procedures (IPP) and operating specs. Other records such as engineering reports, package, process, plant and product qualifications, Failure Analysis Correlation Reports and Material Review Board (MRB) activity reports are controlled via a Technical Information Center.

Table 5. Audit Program

Type of Audit	Scope	Frequency	Audited by	Reason for Audit
Operation Audits	All Wafer Fab, Assembly, Test and QA Operations	Once/Quarter (Twice/Year for Military Programs)	Certified Intel/ Military Auditors	Compliance to Specifications
Mr. Clean Audits	All Facilities	Once/Month	Operation Managers, Facilities	Compliance to Intel Facilities Standards
Vendor Audits	All Material Vendors	Typically, Once/Year	Incoming Inspection and Engineering	Compliance to Specifications

Calibration Lab

All Intel equipment is periodically calibrated using standards traceable to the National Bureau of Standards (NBS). Every major site maintains a calibration lab; remote locations make use of government or privately administered calibration labs.

Analysis Lab

Every major site has an analysis lab. These labs range from highly sophisticated facilities containing a wide variety of analytical equipment (Santa Clara, California; Portland, Oregon) to the product-oriented labs (Folsom, California) to the specialized analytical labs located, for example, at each assembly/test site and in specific fab sites. All labs are tied together through the Analytical Lab Council.

Statistics

Statistical training occurs in three ways: utilization of the Intel University courses provided to all Intel employees; specific and more detailed courses with strategic value presented at individual sites, i.e. SPC; and outside courses that provide expertise not currently available internally. The Q & R statistics functions are provided to all of Intel as a service, much as are analysis lab services, calibration, and document control.

QUALITY AS PART OF INTEL'S CULTURE

While standardization and measurement are important factors in a quality program, there must be a motivation for all personnel to produce a quality product or service. Intel maintains this motivation by funding major programs such as training, Intel Circles (an employee participation awareness program), quality awards for sites and outstanding individual contributors, and the "Mr. Quality" program.

Intel's training programs are extensive. From training line operators at each site, to the M2Q and M3Q middle management quality courses, to the "M5" series of executive courses given by the Intel University, information is made available so that personnel at all levels may improve their performance. There are many courses related to Intel's technical expertise, such as the Intel Technology Course and the Design Microcomputers, in which Intel's top technical personnel take the time to train others. Many thousands of hours are spent by Intel training the new as well as the experienced to produce the products that have made Intel technology synonymous with quality and reliability. All training takes place within Intel's normal working day.

Intel Circles is an adaption of the quality circle concept, in which groups of employees gather locally to discuss problem areas and possible solutions. Activity varies from site to site, but the participation leads to a heightened awareness of quality performance and creative problem-solving.

Rewarding quality performance is an important part of the Intel culture. Each quarter, the Director of Quality and Reliability announces the Executive Staff Quality Award for the highest quality performance of a manufacturing site and an Intel business operation. A secondary award is given to those sites and operations whose performance may not have been the "best" but still surpassed quarterly goals. The competition between sites has been healthy and has spurred efforts which have positively affected the trends shown in Figure 1. This performance takes on personal significance, for both managers' and operators' monthly bonuses are affected by the measured quality relative to goal.

Individual achievement awards are also presented to those employees who have contributed significantly to Intel's quality and reliability improvement. These awards are widely publicized throughout the company. They may go to a physicist for a contribution to EPROM quality or to a line supervisor for improving the quality of shipments to a major account.

Summary of Quality Programs

The Intel Quality and Reliability Department provides a comprehensive program to measure and otherwise assess the quality of Intel product from the time of material purchase to its arrival at the customer's door. Extensive use is made of council and matrix management systems to assure that uniform standards are applied throughout the Intel world. Vast amounts of data are recorded and analyzed in order for Intel to quickly and accurately perceive its quality at any time. The Intel charter of providing a high quality product to its customers is being met by a dedicated effort to daily improvement in the quality ethic (see Figure 1).

RELIABILITY PROGRAMS

Intel manufactures many different products using a variety of fab and assembly technologies. Thus, a wide variety of reliability hazards may be encountered. Some of these are real; such as chips becoming detached from the substrate in a package and consequently failing. Others are imagined or projected into "what if" scenarios. For example, "What would happen if a plastic package were subjected to sudden and severe changes in ambient temperature?" In both the real case and the "what if" case, careful experiments are performed to determine the possible reliability jeopardy of the hazard.

Because of Intel's strong technical background, it has focused on reliability issues and is justly proud of its image as the leader in product reliability. Intel has published extensive technical papers on reliability and is known throughout the industry for its work on soft errors¹, electromigration², charge gain³, oxide breakdown⁴, and a host of other topics. This reputation does not come by accident, for Intel's Quality and Reliability Department has established a variety of procedures and programs to assess, understand, control, and eliminate reliability problems.

The major categories of reliability program management are:

1. Design verification
2. Qualification procedures
3. Manufacturing reliability monitors
4. Fundamental reliability studies

Design Verification

Of key importance is Intel's Q & R interaction with all design groups to ensure that product reliability is designed in. The package reliability and package development engineering personnel ensure that packages will meet strict reliability goals before any package qualification activity is started. Product design engineers work with product reliability engineers to be sure that all products will meet reliability goals. Equally, process development engineers work with the fab and process reliability personnel so that new processes will not violate reliability standards.

In all cases, the Q & R engineers are responsible for signing off appropriate documentation, which ensures that the designs are compatible with reliability goals. The reliability engineers are part of the design team and attend all design review meetings.

Qualification Procedures

Intel qualification procedures are used mainly to ascertain the major characteristics of new technologies or products for introduction to Intel manufacturing, or to evaluate changes to existing technologies or products. In general, all changes require qualification, but the degree of qualification depends on the nature of the change and may range from detailed qualification to simply reviewing engineering data. Qualifications are handled by the appropriate Q & R groups, as shown in Table 6.

1. T.C. May, M. Woods, "A New Physical Model for Soft Errors in Dynamic Memories", Proceedings International Reliability Physics Symposium 16, (1978), p. 33.
2. P.A. Gargini, C. Tseng, M.H. Woods, "Elimination of Silicon Electromigration in Contacts by the Use of an Interposed Barrier Metal", Proceedings International Reliability Physics Symposium 20, (1982), p. 66.
3. M.H. Woods, S. Rosenberg, "EPROM Reliability", Electronics 53, (1980), p. 132.
4. D.L. Crook, "Method of Determining Reliability Screens for Time Dependent Dielectric Breakdown", Proceedings International Reliability Physics Symposium 17, (1979), p. 1.

Table 6. Qualification Activities

Category	Issue	Responsibility
New Plant	Fab Plant Start-up	Fab Q & R
New Plant	Assembly Plant Start-up	Assembly Q & R
New Plant	Test Plant Start-up	Test Q & R
New Package	Totally New Package Technology	Division Q & R
New Product	Totally New Product	Division Q & R
New Fab Process	Totally New Fab Process	Process Reliability
New Equipment	Modified From Existing Equipment	Fab or Assembly Q & R
Parameter Change	Modified Process Parameter	Fab or Assembly Q & R
Product Stepping Change	Modified Product Parameter	Division Q & R
Product Transfer	Fab to Fab	Fab Q & R, Division Q & R
Product Transfer	Assembly to Assembly	Assembly Q & R, Division Q & R
Product Transfer	Test to Test	Division Q & R

As can be seen in Table 6, qualifications take three forms: introduction of new product technology, process or equipment changes or modifications, or startup or transfer to new manufacturing sites. Once qualification begins, it takes considerable time for the qualification samples to be assembled and turned over to the appropriate Q & R group. Therefore, the volume samples available for qualification must be limited, especially since manufacturing volumes are controlled during the qualification phase. Consequently, qualifications do not usually generate DPM-level data, and the only decision made is whether or not the qualification passes the requirements. There is no intention of using process qualification to determine defect levels that would normally require extremely large sample sizes. This issue is taken up by the monitor program discussed later, using production material from which a large database can be generated.

A qualification phase is started by the group, Fab, Product Engineering, Assembly, etc., requesting a modification to an existing technology/equipment/site combination or introduction of a new product/technology/site. This request is usually justified by an improved yield, improved process margin, resolution of a technical issue, lower cost, or improved manufacturing flexibility. Since quals are costly and time-consuming, much thought is given to justifying their need.

The Intel Automotive Integrated Circuit Qualification Plan up to this point reflecting classical qualification concepts is being positioned to accept the most current approved offering of the SAE Qualification Plan. It is Intel's intention to remain in synch with the evolution of the SAE Qualification Plan through continued participation in the SAE Reliability Subcommittee. The SAE Qualification Plan setting the most appropriate PASS/FAIL criteria for automotive class Integrated Circuits relies very heavily on MIL-STD-883 test methods and JEDEC 22A test methods. It is Intel's view that the SAE Power and Temperature Cycle qualification test is particularly relevant to qualification of automotive class Integrated Circuits. It is Intel's further intention to have the Intel Qualification Plan in no way preclude Customer Qualification requirements.

Once the product is given CQ, the product, package, plant and process monitor program starts, and this program continues throughout the life of the product. Generally, since devices or products are the ultimate objects sold by Intel, all process, package and plant qualifications use a specific product as the qualification vehicle whenever possible.

Qualifications are authenticated via reports issued by the appropriate Q & R groups. Qualifications that require division Q & R, as well as package, fab, assembly, or process reliability data, are jointly issued. Due to the large number of qualification issues, the qualification library, administered by the Q & R document control group, is quite extensive.

Reliability Monitors

The reliability monitor program begins where the conditional qualification program ends, at the start-up of limited salable production. Everything that is subject to qualification is considered subject to the monitor program. Some monitors use especially designed test chips, for example, to test for electromigration in metal conductors. However, most monitors use standard production product.

Generally, the product to be used for reliability monitors is gathered from each assembly plant each week, where the product selected is representative of:

1. Each fab process technology
2. Each generic product type
3. Each package technology
4. Each assembly plant

The product is shipped directly to the appropriate Q & R group, which puts the product through a series of electrical, mechanical, thermal, and environmental tests that usually are identical to those used initially for qualifying the product. Most tests are of short duration, but some may extend out to thousands of hours. Each week the test results are evaluated and problems, should they exist, identified.

Each monitor failure is analyzed. If a problem is detected where the failure rate is greater than that considered acceptable, or a reliability problem is suspected, a Material Review Board (MRB) is called. This meeting is attended by appropriate Q & R personnel, scheduling personnel, engineering, and any other affected group. This group reviews the data, decides on disposition of the affected material, decides on appropriate corrective action, and has responsibility for the problem or issue until it is satisfactorily resolved.

Fundamental Studies

Monitors, qualifications, and customer-related issues may generate data that indicate potential reliability problems. While these problems may not have current implications, they could become important as device geometries continue to shrink and chip size increases. Consequently, using the "what if . . ." scenario, Intel reliability engineers have embarked on a number of fundamental studies in order to understand issues sufficiently well to take preventive action.

Examples of such cases have been published widely, in the International Reliability Physics Symposium, for example, and at the Scanning Electron Microscopy Conferences. The most widely publicized case was that of the physics of radioactively-induced soft error failures of dynamic RAMs. Others include such subjects as electromigration, and charge gain in EPROMs.

Generally, a reliability program will be proposed on the basis of either experimental data that indicate the presence of a reliability hazard or on the basis of extrapolation of some trend data to improve proposed new chips or technologies. The program proposal will be reviewed by the appropriate Q & R staff with related engineering groups to decide which programs should be staffed. The programs that are staffed will be reviewed periodically to determine if the experimental results warrant any action on the part of a group developing new technology to prevent future problems. An example of such a program is electromigration, where design rules are generated on the basis of results from periodic electromigration studies.

The fundamental studies undertaken are part of the overall R & D budget of Intel, which amounts to about 12 percent of Intel sales. The programs are reviewed by the Intel Research Council, a group of senior scientists and engineers who review all Intel research activities.

At this time, Intel Q & R has several programs in place, including studies of:

- Stress in packaged chips
- Metal corrosion
- Electromigration
- Hot electrons
- Fault propagation in VLSI random logic
- Soft errors
- Metallization defects
- Silicon crystal defects
- Package strength
- Oxide integrity

CONCLUSION

From these brief descriptions of Intel's commitment to supplying highly reliable products, it can be understood why Intel is considered the world leader in VLSI reliability. Details of the reliability programs will be found throughout this section.

PRODUCT FLOW

Tables 7 through 11 show the product flow through Intel's manufacturing and quality areas.

Table 7. Overview of Q.A./Manufacturing Process Control Operations

Flow	Gates	Qualifications	REL Monitor	Doc't. Control	Calibration	Data Reporting*
Incoming Inspection	Silicon, Piece Parts, Gases Chemicals (Table 10)	All Direct Materials Oper. Audits	Yes	Yes	Yes	DPM, LRR, Vendor Rating
Wafer Fab/ Wafer Sort	Inspections, Process Control (Table 11)	All Materials Processes Technologies Products	Yes	Yes	Yes	Process Yield, Product Yield, LRR
Assembly	Inspections Process Control (Table 12)	All Materials Processes	Yes	Yes	Yes	DPM, LRR, Process Yield, Product Yield
Test	Inspections Process Control (Table 13)	All Products	Yes	Yes	Yes	DPM, LRR, Product Yield
Division/ Customer	Customer Incoming Inspection	By Customer	Yes	N/A	N/A	RMA, FA/CR, Customer Rating

*DPM - Defects per Million
*LRR - Lot Reject Rate

*RMA - Return Material Authorization
*FA/CR - Failure Analysis/Correlation Report

Table 8. Incoming Inspection (MQ)

A. Silicon		
Inspection Type	Inspection	Sample Size
Vendor's Data	To Spec Requirements	All Lots
Visual	High Intensity Lamp	130/Lot Wafers
Dimensional	Diameter, Orientation, Thickness	130/Lot Wafers
Chemical	Oxygen Content	Small Sample
Electrical	Resistivity, Type, Lifetime	130/Lot Wafers

NOTE:

Other criteria, such as dislocation density or flatness are checked periodically for each manufacturer.

B. Piece Parts		
Inspection Type Leadframes	Inspection	Sample Size
Vendor's Data	To Requirements of Purchase Spec.	All Lots
Visual	To Requirements of Purchase Spec.	52/Lot
Mechanical	Coplanarity	45/Lot
	Die Pad Twist	45/Lot
	Lead Twist	45/Lot
	Metal Thickness	45/Lot
	Camber	5 Strips/Lot
	Bow	45/Lot
Functional	Coil Set	45/Lot
	Die Attach	45/Lot
	Extended Heat	45/Lot
	Tape (peel) Test	45/Lot
	Lead Fatigue	45/Lot
	Bond Pull	136 Wires/Lot

NOTE:

A Plastic Package Leadframe is used as an example of piece part incoming inspection. All piece parts are subject to incoming inspection requirements.

Table 9. Wafer Fabrication and Wafer Sort Process Control Inspections and Monitors

Process Step	Process Control Inspection	Frequency	Process Control Monitor	Frequency
Oxidation	Oxide Thickness	All Lots	Low Mag. Visual C/V Test on Tubes	Once/Shift Once/3 Days
Diffusion	Oxide Thickness Sheet p 1 X Visual	All Lots All Lots All Wafers	Low Mag. Visual C/V Test on Tubes	Once/Shift Once/3 Days
Lithography	Low, High Mag. Visual	All Lots	Masking Visuals	15% of Lots
Thin Film	Metal Thickness	All Lots	C/V Test Low Mag. Visual	Once/Shift Once/Shift
Ion Implant	Sheet p	All Lots		
Low Temp. Oxide	Thickness Reflectivity Low Mag. Vis.	All Lots	Visual	Twice/Week
E-Test	Controlled Elec. Tape	All Lots		
Wafer Sort	Controlled Elec. Tape	All Lots	Tapes Approved by Q & R Revision Cont'l Visual (MFG)	All Tapes All Tapes All Lots
Controlled Spec's.			Line Audit	Once/Day
Calibration			Line Audit	Once/Day

Table 10. Assembly Process Control

Assembly Process Control Monitors (Tin-Plated Cerdip Mfg. Flow)				
Process Step	Process Control Gate	Frequency	Process Control Monitor	Frequency
Wafer Saw, Break, Wash	DI H ₂ O Resistivity	1X/Shift	Visual	2X/Shift/Operator
2nd Op	—	—	Visual	All Lots
Frame Attach	—	1X/Shift/Machine	Visual	2X/Shift/Machine
Die Attach	Heater Block T, N ₂ Flow Rate	2X/Shift/Machine	Visual	4X/Shift/Machine
Bond	Bond Pull	All Lots	Visual	2X/Shift/Machine
Pre-seal Visual	—	—	Visual	All Lots
Seal	Furnace Profile	1X/Furnace/Day	—	—
Post-Seal Visual	—	—	Visual	2X/Shift/Furnace
Bottomside Mark	Permanency	All Lots	Visual	2X/Shift/Operator
Tin Plate	Bath Composition	2X/Shift/Bath	Visual	2X/Shift/Both
Trim	Settings	1X/Shift/Machine	Visual	2X/Shift
Outgoing QA	Hermeticity Centrifuge-Open/Short Acoustic/PIND Lead Fatigue Open/Short	All Lots All Lots All Lots All Lots	Visual	All Lots

Table 11. Test, Mark, Pack Final Process Control Environmental Monitor

Process Step	Process Control Gate	Frequency	Process Control Monitor	Frequency
Electrical Test	Equipment Setup	All Product, All Lots	Proper Test Tape	All Changes
Mark	Ink Control, Data Code, Storage	All Batches of Ink	Visual	All Lots
Pack	Pack Check	All Lots	—	—
Final QA	Elec. Test Hermeticity Mark Permanency Solderability	All Lots All Non-Plastic Lots All Lots All Lots	Visual — — —	All Lots — — —
Final QA Preclearance	Unit Count, Labeling Invoice Check	All Lots	—	—
Reliability Monitor	—	—	1000-Hr Dynamic Burn-in	Monthly Each Technology, per Reliability Schedule

These flows are rigidly followed for standard product. The flow is modified only by customer specification or qualified Intel engineering change and is controlled by the Intel specification system.

The quality measurement systems within these areas must also be identical if visibility and comparison of quality are to have any meaning. For example, all sites issue "DPM" (Defects Per Million) reports based on quality assurance key inspection points (see Tables 7 through 11). At each inspection point, the definition of DPM is:

$$\text{DPM} = \left(\frac{\text{number of defective units}}{\text{total number of units sampled}} \right) \times 10^6$$

Thus, whether in Assembly or Test, DPM as a quality indicator has but one meaning. All such indicators are standardized throughout the Intel components world.

LIFE CYCLE OF A PRODUCT

Quality and Reliability involvement in the product life cycle is defined below. Note that the Q & R organization is involved from the birth of a product through its useful lifetime in the field. The 2764, Intel's UV Erasable 64K PROM (EPROM) is used as an example of the 16 steps required to complete a life cycle:

1. PRODUCT DEFINITION

Based on Marketing inputs, definition of the proposed product was performed by a Strategic Business Segment (SBS) Committee. This committee is composed of representatives of Marketing, Manufacturing, Technology Development, Engineering, and Quality and Reliability. This group proposed the development of the 2764 and defined the potential Return On Investment (ROI).

2. TECHNOLOGY DEFINITION

Proposed by Technology Development (TD), the technology definition required to generate a 2764 product was based on both experience and need for state-of-the-art growth. The 2764 was designed on the H-MOS-E, "wafer stepper" process. This decision was also coordinated through the SBS.

3. TARGET SPECIFICATION

The target goals of the product proposed by TD and Design Engineering were coordinated through the SBS. Quality and Reliability, Marketing and Manufacturing must review and sign off the target specification. For the 2764, Reliability Engineering worked with Design Engineering to provide "margin testing" modes to ensure reliability, manufacturability, and testability commensurate with Intel Q and R targets and customer needs.

4. LAYOUT/DESIGN RULES

Based on experience and product simulation, design rules for a technology were developed by TD, verified by Process Reliability, and implemented in design. By following these rules, known failure modes were eliminated in the new product. The 2764 employed the experience learned on earlier EPROM products, the 2716 and 2732, which had wider design margins for the cell memory.

5. COMPUTER-AIDED DESIGN (CAD)

The computer not only produced the magnetic tapes from which the masks were made but tested the design to assure that the target specification was met and that no design rules were violated. The 2764 was subjected to an intensive design, circuit, and layout review by Design Engineering, Manufacturing, and Quality and Reliability.

6. TAPE-OUT/MASK

The magnetic tape generated by the CAD group was translated to the masks utilized in the fabrication of the silicon wafers. All masks which were purchased from outside vendors were subject to rigid incoming inspection by the Materials Quality section of Q & R.

7. FIRST SILICON

The first silicon ("A Stepping") of the 2764 was functionally successful. As a result of intensive teamwork in the design stage, the characteristics of the first wafers could be analyzed against the product goals.

8. DESIGN/PERFORMANCE REVIEW

Design Engineering, Manufacturing, and Quality and Reliability Engineering performed tests on the functionally operable circuits to check test margins, cell design, appearance of known failure modes and program-erase cycles. Worst-case conditions (voltage, temperature) helped to identify and characterize potential problem areas, resulting in design changes and implementation of reliability screen tests.

9. RELIABILITY INVESTIGATIONS

The analysis was not confined to the single 2764 product. Knowledge gathered on the 2764 was extended to other products in the form of modifications, test screens, and device monitors to further enhance overall product quality and reliability.

10. REDESIGN

The 2764 experienced some programming anomalies in first silicon. Reliability Engineering evaluated the problem and was able to suggest design changes based on the earlier 2732 technology. The resulting design changes were reviewed, approved, and implemented.

11. "B STEP"

The new masks were inspected at incoming inspection and wafers were processed. The circuit went into the product validation/qualification cycle.

12. ENGINEERING VALIDATION

A limited number of internal and external applications sites were supplied with sample devices. Since over 90% of Intel's own product types are used in Intel systems, it is critical that the system design areas feed back their experience early in the product cycle.

13. QUALITY VALIDATION

Three hundred devices were tested to specifications (no guardband). They were tested to all corners of the specification, utilizing standard programming to various patterns. This elaborate matrix resulted in a more effective test program, in identification of a new defect mode, and in development of a stress test to eliminate the mode from device production. The result: the first EPROM driven to a defect level of less than 1000 DPM on programming.

The EPROM had to pass a rigid, documented qualification program before reaching the marketplace. This program included the goal for the 2764 of 99.9% programming yield or less than 1000 DPM. The qualification testing included the following sequence on a minimum of five wafer lots:

125°C burn-in	168 hours
125°C lifetest	2000 hours
125°C HVELT	1000 hours
Low temperature lifetest	1000 hours
250°C storage	1000 hours
Temperature cycle -65°C to +150°C	200 cycles
Thermal shock -65°C to +150°C	200 cycles

Test pattern study
Program/erase cycling
System verification

Each test had a specific reliability goal to meet. The 2764 successfully passed all tests.

14. QUALITY TASK FORCE

A Quality Task Force was formed to reinforce the commitment to production of a product of the highest quality. Areas such as mark, visual appearance, tin plate quality and assembly quality were continuously evaluated during the first six months of product life. Defects-per-million (DPM) statistics were developed to assure that all goals were being met or exceeded.

15. QUALITY AND RELIABILITY MONITORS

As part of any Intel product cycle, both quality and reliability monitors are established to continuously sample products directly from the production line. Package types and technologies are subjected to these monitors, and results are made known to the various responsible engineering, manufacturing and quality groups. The quality monitors include:

- external visual inspection
- fine/gross leak test (hermetic packages)
- lid torque test
- temperature/humidity test (for plastic packages)
- internal visual inspection
- temperature cycle test
- thermal shock test
- steam stress test

The reliability monitors include 48-hour dynamic burn-in at 125°C, 1000-hour dynamic lifetest at 125°C, and 250°C storage. The 2764 had a special reliability monitor in which devices were subjected to data sheet test conditions with various patterns programmed.

16. CUSTOMER FEEDBACK

Intel customers provide the ultimate testimony to Intel efforts. One large U.S. computer systems manufacturer has consistently measured 50 DPM or less on the 2764. Another system manufacturer measured zero rejects in 50,000 devices inspected. The average for all Intel mature and new products is below 1000 DPM total.

CONCLUSION

This example of the development, production, and field monitoring of a new product illustrates Intel's approach to organization and management. The flow of the product through the various stages has been perfected to ensure consistently high quality and reliability. It should be noted that as product quality and reliability continue to improve, inspection sample plans will be modified as more reliance is placed on process control activities. This is consistent with Intel plans to purchase and supply zero DPM products.

The engineer constantly evaluates the "test structures" on the technology for reliability problems. The reliability engineer is responsible for ensuring that the design is consistent with the reliability goals.

CHAPTER 3 RELIABILITY

Product Design Phase

The reliability of Intel devices has been unmatched since Intel established its product line in 1969 with the invention of the DRAM. The subsequent years which saw Intel introduce first the microprocessor chip and then the EPROM chip has been a period of intense awareness for Intel of the issues contributing to outstanding product reliability. In this period Intel has made major contributions to enhancing the science of Integrated Circuit Reliability. Tim May's work in characterizing alpha particle-based soft errors in DRAMs would, of course, be an excellent example of this contribution.

Intel's formula for success is quite simple: design reliability into the chip, manufacture it under stringent controls, and have a system of product test that is so thorough that the product delivered to the field is defect free. This formula is not always easy to adhere to, but adhere to it, Intel does.

In this chapter the four aspects of Intel's Integrated Circuit reliability are discussed:

- Design for Reliability
- Process Reliability
- Product Reliability
- Package Reliability

The chapter closes with a brief discussion of the reliability monitor program.

DESIGN FOR RELIABILITY

Introduction

The dawn of the design—both for the product and the process technology—is the point at which “design in” for reliability begins. It is at this point that the Q & R Reliability Engineers enter the new design project and begin the task of interacting with the design team and the design such that the product goals of the design (product and process) and the reliability goals are simultaneously met. The mechanisms needed for reliability “design in” more often than not can only be installed at design onset.

Product and Process Definition Phase

A product definition requires a consensus of Design, Manufacturing and Q & R. A Q & R representative is a part of the committee responsible for defining the “target specification” of a product. This representative ensures that:

1. Features necessary for quality and reliability are included in the product definition. Examples of such features will be given later.
2. The correct reliability and quality goals are specified. For example, the acceptable soft error rate for memory products is defined. Similarly, the acceptable defect density for a semiconductor processing technology is specified.

The Q & R representative is required to sign off the final “target specification” document.

Process Development Stage

A process reliability engineer is assigned to ensure that a new semiconductor processing technology will meet its reliability and quality goals. This engineer takes an active part throughout the semiconductor process development cycle, meeting with the process development engineers on a regular basis and keeping abreast of their progress.

The engineer constantly evaluates the "test structures" on the technology for reliability problems. The reliability engineer is responsible for ensuring that the design rules used by the chip designers are consistent with the reliability goals.

Product Design Phase

The product designers use the design rules established by the process engineers to lay out the product. Due to the ever-increasing complexity of the products, it is becoming impossible to manually lay out chips without error. Automated chip layout is, quite simply, what has to be done to effect an efficient and essentially error-free chip layout. In the matter of chip layout errors Intel has pioneered the use of several computer tools to identify layout design errors. Such tools are a Design Rule Checking (DRC) program to catch layout design rule violations and a Continuity Verification System (CVS) to catch circuit "hook-up" errors (improper circuit connections). In addition, Intel design engineers use circuit and logic simulation programs to predict product performance.

The reliability engineer reviews final chip layout and verifies that the reliability requirements are satisfied.

Features to Ensure Quality and Reliability

At Intel all known semiconductor failure mechanisms are extensively characterized and modeled. Based upon this understanding, one of the following approaches is used during design to eliminate each failure mechanism.

LARGE DESIGN MARGINS

Most potential reliability problems can be designed out by modifying the design to accommodate large margins—voltage, timing and, possibly, temperature. The margin has to go beyond the statistical variation of the production process. Large design margins are synonymous with better reliability. Care is taken during design rule definition and characterization to ensure the largest possible margins. Once the product goes into large-volume manufacturing, the margins are monitored on a regular basis to ensure no dangerous and abnormal drift of the process.

MARGIN MONITORING CAPABILITY

Unfortunately, large margins are not a practical solution for all failure mechanisms. Due to cost and performance trade-offs, smaller margins may become unavoidable. For such failure mechanisms Intel measures the margin for every individual unit, and units which have poorer margins are screened out. Reliability Engineering works closely with design engineers to modify the design, so that the margin can be measured on the final product. This often requires significant design ingenuity. For example, on dynamic memory products, which can have significant soft error rates, a special "dummy" transistor indicates the magnitude of soft error margin. A discussion of program margin testing for EPROM is given in an earlier section of this handbook. The reliability engineer ensures that the proper margin testing capability is implemented.

OVERSTRESSING TO KILL WEAK PRODUCTS

Weak gate oxides (those exhibiting low breakdown voltage due to the presence of defects) are screened out by deliberately overstressing the product. The oxide stress test requires the product to be operated at higher-than-rated electric fields and operating voltages. To effectively execute the oxide stress test certain design precautions may be necessary. For example, on-chip voltage regulator circuits clamp certain nodes to fixed voltages. Such nodes cannot be stressed (because the voltage is clamped). To prevent this, an "override" capability is designed into the voltage regulators. This mechanism allows the correct implementation of the oxide stress test.

Once the product is determined to be functional, it undergoes an extensive characterization and evaluation. The design engineer formulates a characterization and evaluation plan. Other organizations, such as Manufacturing, Application Engineering, and Reliability Engineering, approve the characterization plan and also add their requirements to the plan. An objective of the characterization is to ensure that the product will have adequate margins relative to the data-sheet parameters. Any marginalities and problem areas are identified.

A typical evaluation consists of a "schmoo" plot, Figure 1. A schmoo plot checks that the complex interaction of inputs does not lead to subtle problems.

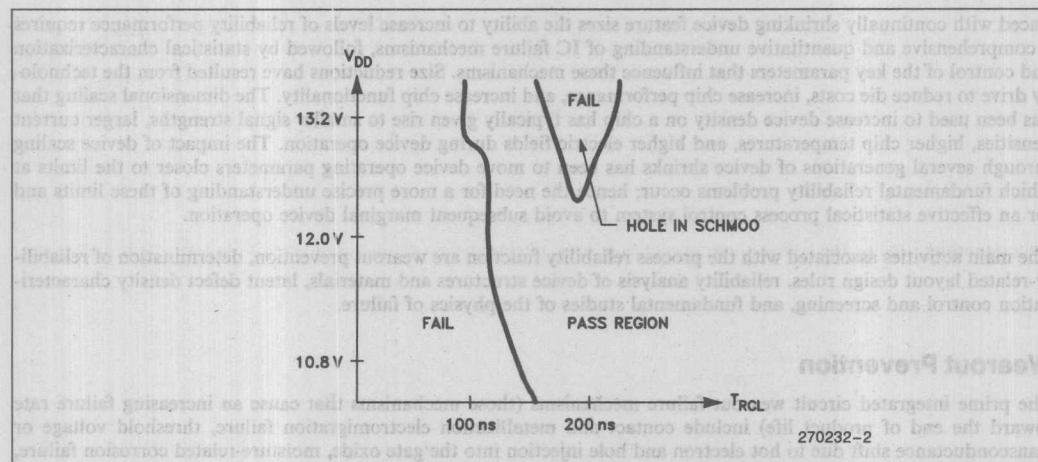


Figure 1. "Schmoo" Plot of a Memory Device, Showing Combinations of V_{CC} and t_{RCL} for Which Correct Device Functioning is Obtained

As seen in Figure 1, a subtle interaction causes a product malfunction only at certain combinations of V_{CC} and t_{RCL} (delay between two clocks). A schmoo plot uncovers this marginality, which would have passed unnoticed if only a few combinations of t_{RCL} and V_{CC} had been tested.

Another objective of the characterization phase is to define the impact of the manufacturing process variables on product margins. Special characterization lots are generated, where the process variables are deliberately run at the extremes of the control limits. The impact of these process variations on the product performance is characterized and documented, as illustrated in Table 1.

Table 1. Impact of "Poly Feature Size" and "Gate Oxide Thickness" on the Access Time of a 16K RAM

GATE OXIDE THICKNESS	POLY	FEATURE	SIZE
	Wide	Typical	Narrow
Thick	197 ns	171 ns	157 ns
Typical	183 ns	159 ns	141 ns
Thin	167 ns	155 ns	137 ns

Knowledge of the impact of the manufacturing process variables on product performance makes it easier for Manufacturing to ensure the largest performance margins. If a manufacturing variable can have a major impact on the reliability margins of a product, special characterization lots are generated to study the impact in greater detail. For example, feature size on diffusion masks and the gate oxide thickness significantly affect the soft error margin for memory products. Thus, a process split with various combinations of diffusion feature size and oxide thickness is generated. The soft error margin is characterized for each combination.

PROCESS RELIABILITY

Introduction

The Process Reliability group ensures that the components of new technologies (process steps, device structures, materials, equipment, design rules, circuit functionality) by which integrated circuits are made are inherently reliable. That is, this group ensures that no intrinsic wearout mechanisms are built into the technology, that defect densities are reduced to sufficiently low levels to allow efficient and economical product manufacture, and that layout and circuit design rules guarantee safe operation throughout the lifetime of the customer's products.

Faced with continually shrinking device feature sizes the ability to increase levels of reliability performance requires a comprehensive and quantitative understanding of IC failure mechanisms, followed by statistical characterization and control of the key parameters that influence these mechanisms. Size reductions have resulted from the technology drive to reduce die costs, increase chip performance, and increase chip functionality. The dimensional scaling that has been used to increase device density on a chip has typically given rise to smaller signal strengths, larger current densities, higher chip temperatures, and higher electric fields during device operation. The impact of device scaling through several generations of device shrinks has been to move device operating parameters closer to the limits at which fundamental reliability problems occur; hence the need for a more precise understanding of these limits and for an effective statistical process control system to avoid subsequent marginal device operation.

The main activities associated with the process reliability function are wearout prevention, determination of reliability-related layout design rules, reliability analysis of device structures and materials, latent defect density characterization control and screening, and fundamental studies of the physics of failure.

Wearout Prevention

The prime integrated circuit wearout failure mechanisms (those mechanisms that cause an increasing failure rate toward the end of product life) include contact and metallization electromigration failure, threshold voltage or transconductance shift due to hot electron and hole injection into the gate oxide, moisture-related corrosion failure, oxide wearout, and E²PROM program/erase cycling-induced oxide failures.

Wearout mechanisms, because of their very nature, are difficult to test on the actual integrated circuits, since these devices do not often lend themselves to proper operation under highly life-accelerating voltage, current or temperature conditions. Therefore, many wearout mechanisms are characterized on specially designed test patterns operated under highly accelerated conditions. A great deal of understanding of the reliability physics and actual circuit layout and operating conditions is then needed to relate these results to those that would be obtained by an IC during normal device operation. Since wearout prevention can often have a major impact on the design rules for device layout, these studies are among the first to be undertaken when new technology undergoes initial development and characterization.

Electromigration

Electromigration^{1, 2-4} is the directional modification of the motion of the metallic ions of the integrated circuit conductors and interconnects due to momentum exchange from the current carrying electrons. A positive divergence of the metallic ion flux leads to void formation in the conductors and ultimately to open circuits as shown in Figure 2. A negative divergence of the ion flux at a point in the conductor can lead to hillock formation or a build-up of metal in local regions and possibly to short circuits between adjacent or overlying conductors.² Contacts of metallization to silicon can fail not only by metallization voiding of the contact side wall but also by the continuing process of electromigration of silicon out of the contact and back diffusion of the aluminum into the void until the underlying p-n junction is shorted out. Figure 3 shows the voiding of the one contact, and the subsequent hillock growth in the next contact is shown in Figure 4.

1. P. A. Gargini, C. Tseng, M. H. Woods, "Elimination of Silicon Electromigration in Contacts by the Use of an Interposed Barrier Metal", Proceedings International Reliability Physics Symposium 20, (1982), p. 66.
2. I. A. Blech, E. S. Meieran, "Electromigration in Thin Al Films", Journal of Applied Physics 40, (1969), p. 485.
3. S. Vaidya and A. K. Sinha, "Electromigration Induced Leakage at Shallow Junction Contacts Metallized with Aluminum/Poly-Silicon", Proceedings International Reliability Physics Symposium, 20 (1982), p. 50.
4. I. A. Blech, "Copper Electromigration in Aluminum", Journal of Applied Physics 48, (1977), p. 473.

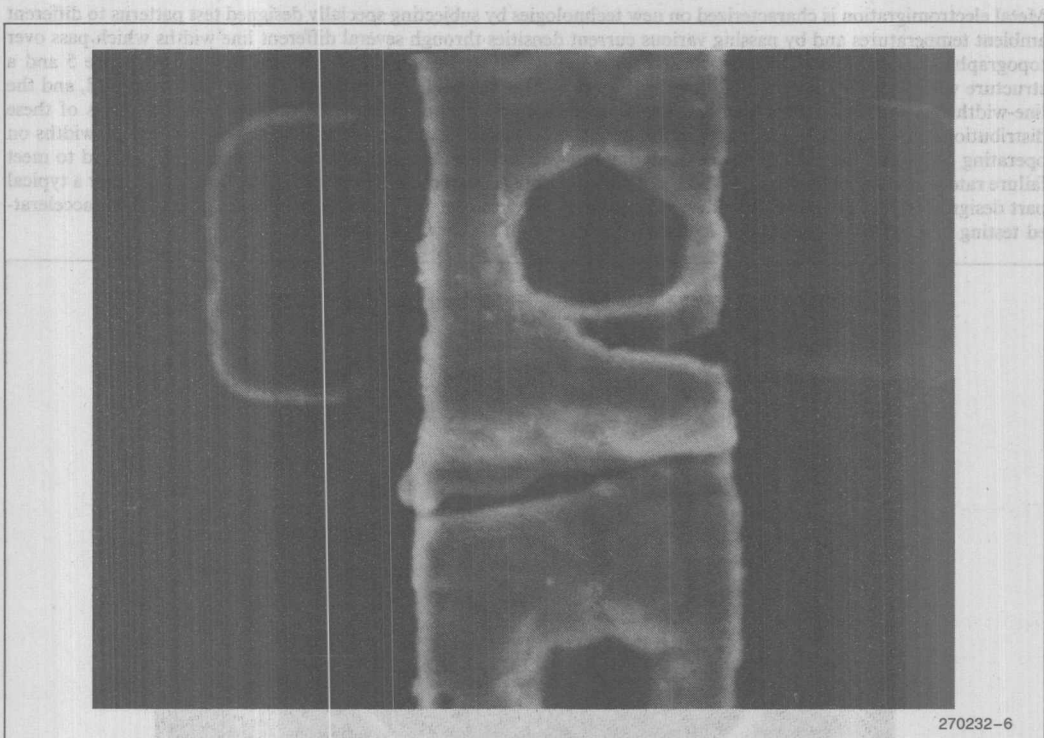


Figure 2. Electromigration Failure

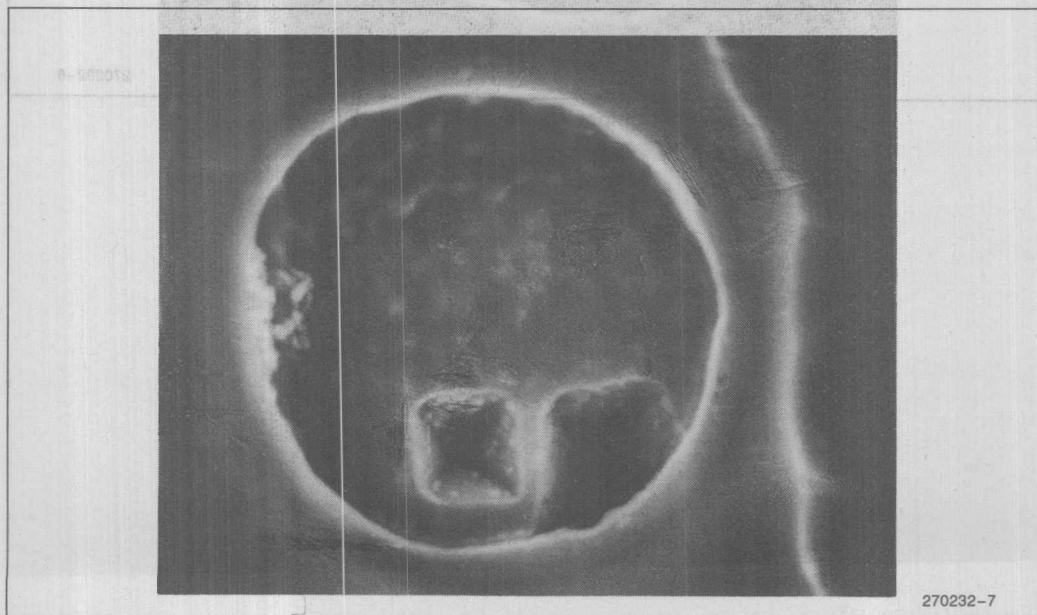


Figure 3. 5th Contact in a Chain With Voiding

Metal electromigration is characterized on new technologies by subjecting specially designed test patterns to different ambient temperatures and by passing various current densities through several different line widths which pass over topographical steps. The SEM cross-section for a metal line with good step coverage is shown in Figure 5 and a structure with poor step coverage is shown in Figure 6. The various failure distributions are then measured, and the line-width dependence, current density, and temperature acceleration factors are derived from the shifts of these distributions with time. The electromigration design rules which limit the current allowed in various line widths on operating chips as a function of device operating temperature are then determined. These rules are defined to meet failure rate goals for the first 100,000 hours of operating life. The fraction failing due to electromigration for a typical part designed to the electromigration design rules should be less than 0.1%. These results are predicted for accelerated testing of a small sample using a log-normal distribution as shown in Figure 7.

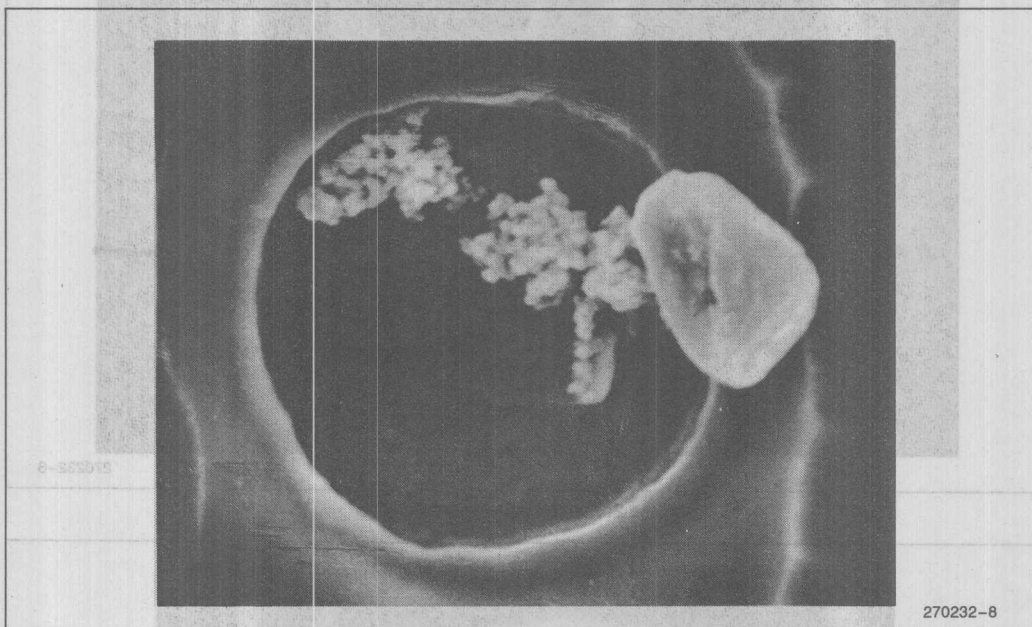


Figure 4. 6th Contact in a Chain With Hillock Growth

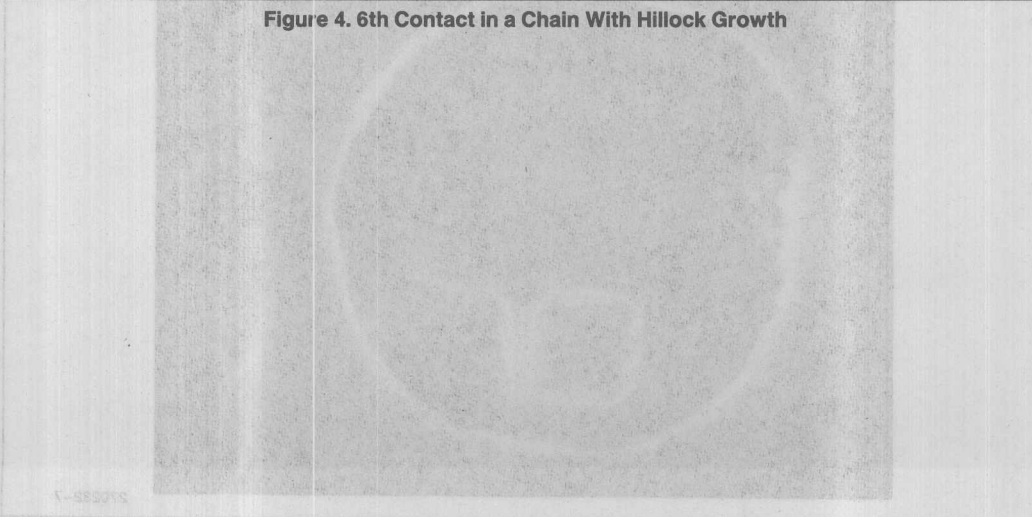


Figure 3. 5th Contact in a Chain With Voiding

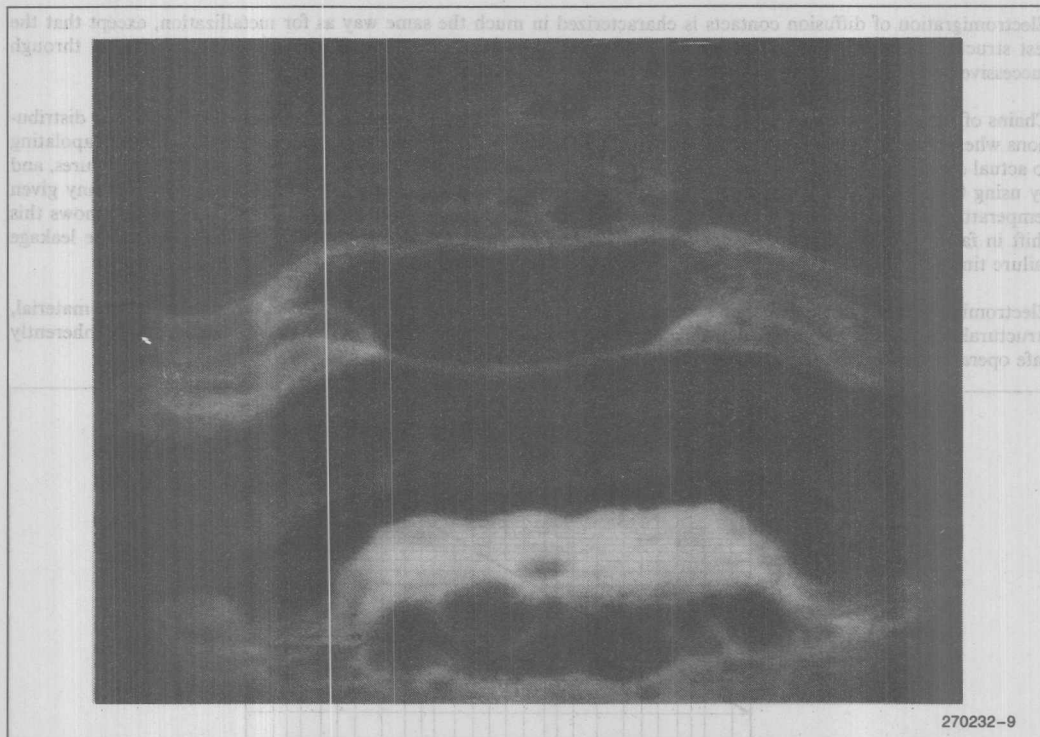


Figure 5. Metal Line With Good Step Coverage

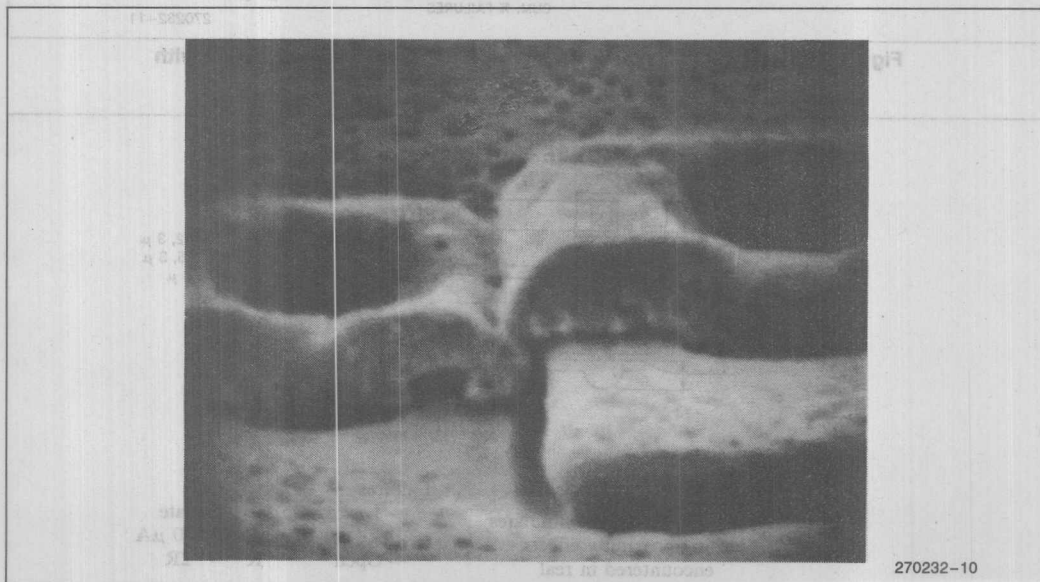


Figure 6. Metal Line With Poor Step Coverage

Electromigration of diffusion contacts is characterized in much the same way as for metallization, except that the test structures consist of chains of metal-to-diffusion contacts. During stress, current passes from metal through successive contacts to diffusion. A schematic cross-section of the test contact chain is shown in Figure 8.

Chains of contacts of various geometries are characterized for both leakage and open circuit failure mode distributions when subjected to different stress currents and temperatures. The design rules are determined by extrapolating to actual operating temperatures using 0.9eV (the silicon electromigration activation energy) for leakage failures, and by using 0.5eV (the aluminum electromigration activation energy) for open failures. The design rule at any given temperature is the lesser of the two currents which give safe operation for each failure mode. Figure 9 shows this shift in failure mode for two different junction depths. The junction depth has a dramatic impact on the leakage failure time.

Electromigration is rendered neither a reliability nor a performance problem by first optimizing the metal material, structural and geometrical properties, then by developing and implementing the design rules which ensure inherently safe operation, and finally by process control of key metallization parameters.

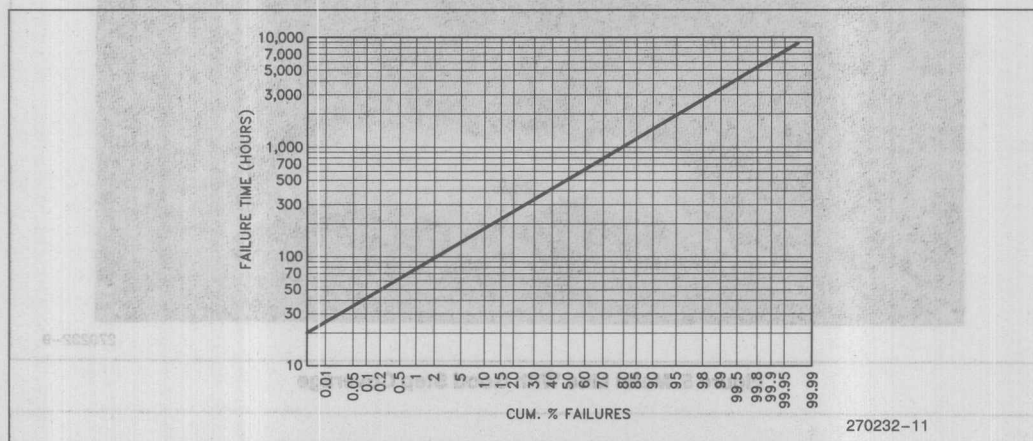


Figure 7. Electromigration Failure Rate Versus Time for a Distribution with
MTTF = 200 Years and Sigma = 1.0

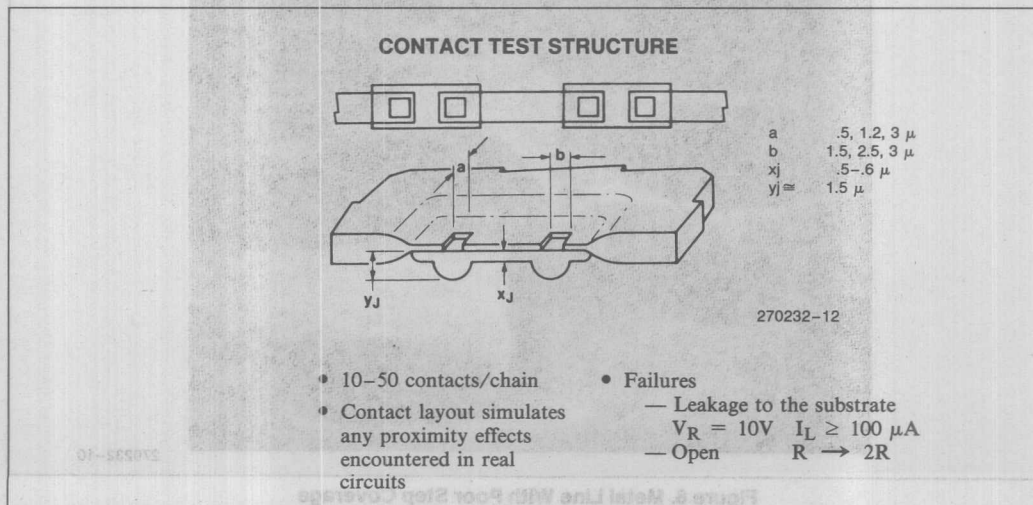


Figure 8. Contact Test Structure Used for Electromigration Tests

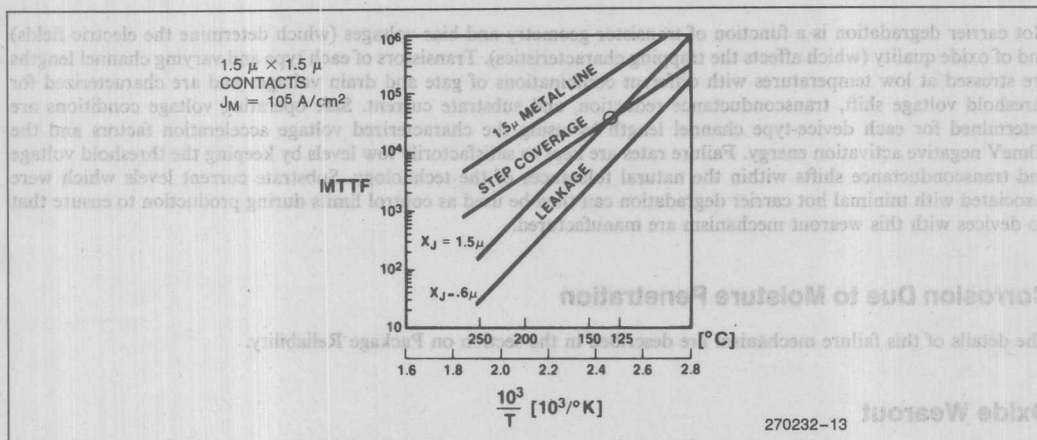


Figure 9. Leakage and Open Failure for 1.5 m x 1.5 m Contacts When a Stress Current of 10⁵ A/cm² is Used in the Metal Lines

Hot Carrier Injection

Hot carrier conditions occur in most transistors when the lateral electric field in the drain depletion region or in the channel becomes so large that the conducting electrons or holes gain energy from the electric field faster than they can lose it back to the silicon lattice via optical photon emission. These highly energetic carriers are termed hot carriers because they have a much greater distribution of energies than the silicon temperature would predict.⁵ As these carriers take on more energy from the electric field, they first gain enough energy (1.5eV) to impact-ionize and create hole-electron pairs, which results in a measurable substrate current consisting of carriers created with opposite conductivity type to those in the channel. If the hot carriers further gain enough energy (3-4eV) and through collisions obtain the proper momentum, they can surmount the Si-SiO₂ energy barrier and be injected into the overlying SiO₂. Hot carriers can also tunnel quantum mechanically directly into the oxide at lower energies.

Depending on the sign of the carrier's charge and the direction of the electric field in the oxide at the point of injection, the electron or hole is either drawn toward the gate or driven back toward the channel, Figure 10. The direction of the oxide field above the channel is a function of both the gate and the drain voltage. While in the oxide, a percentage of the injected carriers will be trapped. These trapped charges can lead to transistor parameter shifts and subsequently to device malfunction. For example, electrons trapped above the channel of an n-type transistor can locally raise the threshold voltage and reduce the drain current. Holes trapped near the interface can lead to the formation of interface states which reduce the differential transconductance of a transistor. The reduced gain can subsequently lead to device failure.

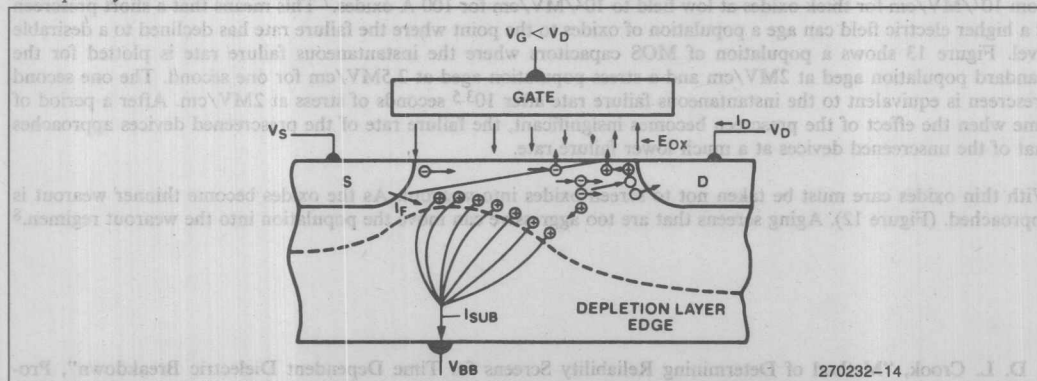


Figure 10. Impact Ionization and Hot Carrier Injection

5. C. Hu, "Hot Electron Effects in MOSFETS", IDEM Tech. Digest, International Electron Device Meeting, Wash. D.C. (1983), p. 176.

Hot carrier degradation is a function of transistor geometry and bias voltages (which determine the electric fields) and of oxide quality (which affects the trapping characteristics). Transistors of each type and varying channel lengths are stressed at low temperatures with different combinations of gate and drain voltages and are characterized for threshold voltage shift, transconductance reduction, and substrate current. Safe operating voltage conditions are determined for each device-type channel length by using the characterized voltage acceleration factors and the 60meV negative activation energy. Failure rates are kept to satisfactorily low levels by keeping the threshold voltage and transconductance shifts within the natural tolerances of the technology. Substrate current levels which were associated with minimal hot carrier degradation can then be used as control limits during production to ensure that no devices with this wearout mechanism are manufactured.

Corrosion Due to Moisture Penetration

The details of this failure mechanism are described in the section on Package Reliability.

Oxide Wearout

Oxide defect-related failures typically occur early in product life and give rise to a declining failure rate in the infant mortality regime of the reliability failure distribution. As such, devices with oxide defects are screenable by high-voltage stress or burn-in.

Oxide wearout is designated as such because of an increasing failure rate due to this mechanism at the end of the reliability life curve. The failure mechanism governing oxide wearout may be exactly the same as that controlling some of the defect-related failures, except wearout occurs in a much more uniform fashion on non-defective oxides.

Oxide breakdown is preceded by the build-up of positive charge in the oxide near the silicon interface, which enhances the injection of electrons from the silicon into the SiO_2 . Excessive field and current in the SiO_2 can also lead to the generation of electron traps and electron trapping in the oxide. This process continues until the oxide fails at the weakest spot, at which time the charge stored in the gate capacitor discharges through this weak location. The associated localized Joule heating vaporizes the SiO_2 and silicon and ruptures the capacitor. The oxide current is described mathematically by the Fowler-Nordheim equation and as such is exponentially dependent on electric field. Therefore, for constant supply voltage the MTTF is exponentially dependent on oxide thickness.

Both defect-related and wearout-type oxide breakdown are characterized on each process by running constant field time-dependent dielectric breakdown (TDDB) and ramped voltage breakdown tests on MOS structures that are unambiguously related to the length over the diffusion/field oxide edge, the length of gate oxide edge, or the gate oxide area. Oxide breakdown is highly accelerated by electric fields and weakly accelerated by temperature.⁶ The electric field acceleration can be calculated from time-dependent dielectric breakdown data shown in Figure 11. As the oxide thickness decreases and electric field increases, the field acceleration factor has been shown to decrease from $10^7/\text{MV}/\text{cm}$ for thick oxides at low field to $10^2/\text{MV}/\text{cm}$ for 100 \AA oxides.⁷ This means that a short prescreen at a higher electric field can age a population of oxides to the point where the failure rate has declined to a desirable level. Figure 13 shows a population of MOS capacitors where the instantaneous failure rate is plotted for the standard population aged at $2\text{MV}/\text{cm}$ and a stress population aged at $2.5\text{MV}/\text{cm}$ for one second. The one second prescreen is equivalent to the instantaneous failure rate after $10^{3.5}$ seconds of stress at $2\text{MV}/\text{cm}$. After a period of time when the effect of the prescreen becomes insignificant, the failure rate of the prescreened devices approaches that of the unscreened devices at a much lower failure rate.

With thin oxides care must be taken not to screen oxides into wearout. As the oxides become thinner wearout is approached. (Figure 12). Aging screens that are too aggressive can move the population into the wearout regimen.⁸

6. D. L. Crook, "Method of Determining Reliability Screens for Time Dependent Dielectric Breakdown", Proceedings International Reliability Physics Symposium 17, (1979), p. 1.

7. J. W. McPherson, D. A. Bagler, "Acceleration Factors for Thin Gate Oxide Stressing", Proceedings International Reliability Physics Symposium, (1985), pp. 1-5.

8. W. Meyer, D. Crooks, "A Non-Aging Screen to Prevent Wearout of Ultra-Thin Dielectrics", Proceedings International Reliability Physics Symposium, (1985), pp. 6-10.

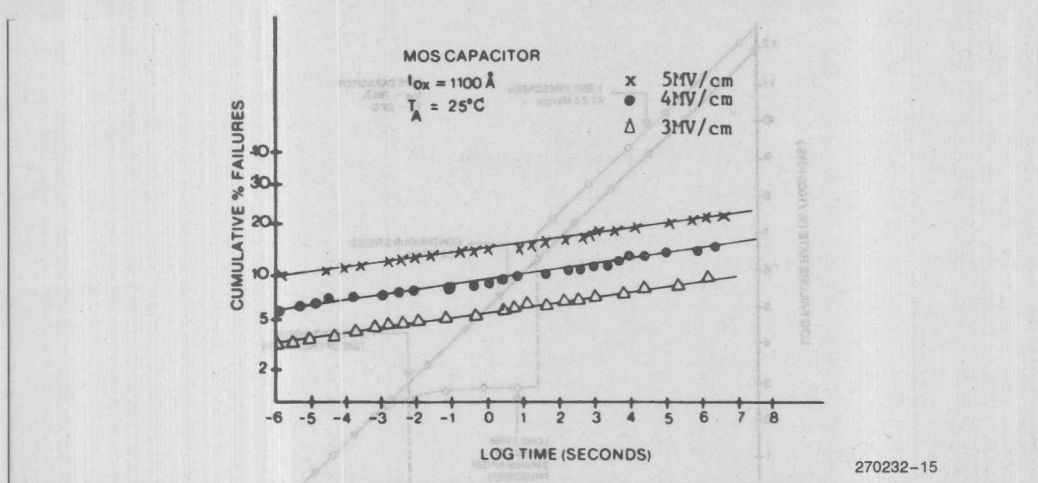


Figure 11. Time Dependent Dielectric Breakdown Data for Three Different Electric Fields

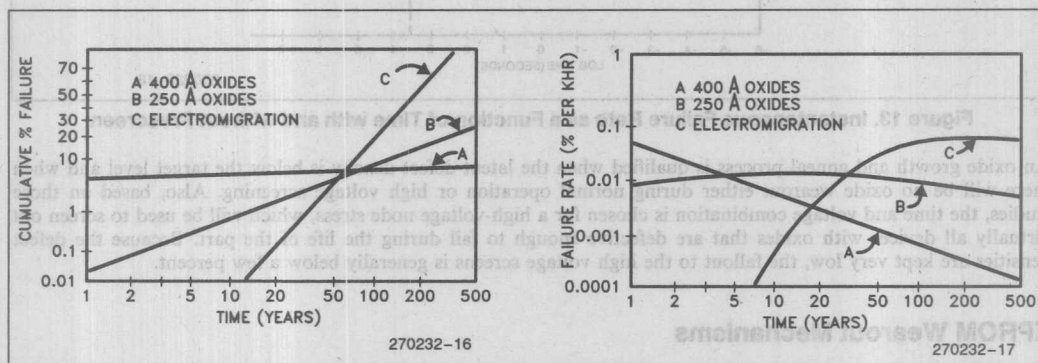


Figure 12. Examples of 3 lognormal failure distributions. Two oxide distributions are compared to a typical electromigration distribution. a) Cumulative failures vs. time as measured at accelerated conditions and extrapolated to normal operating conditions. b) Predicted time-dependent failure rate for the same distributions.

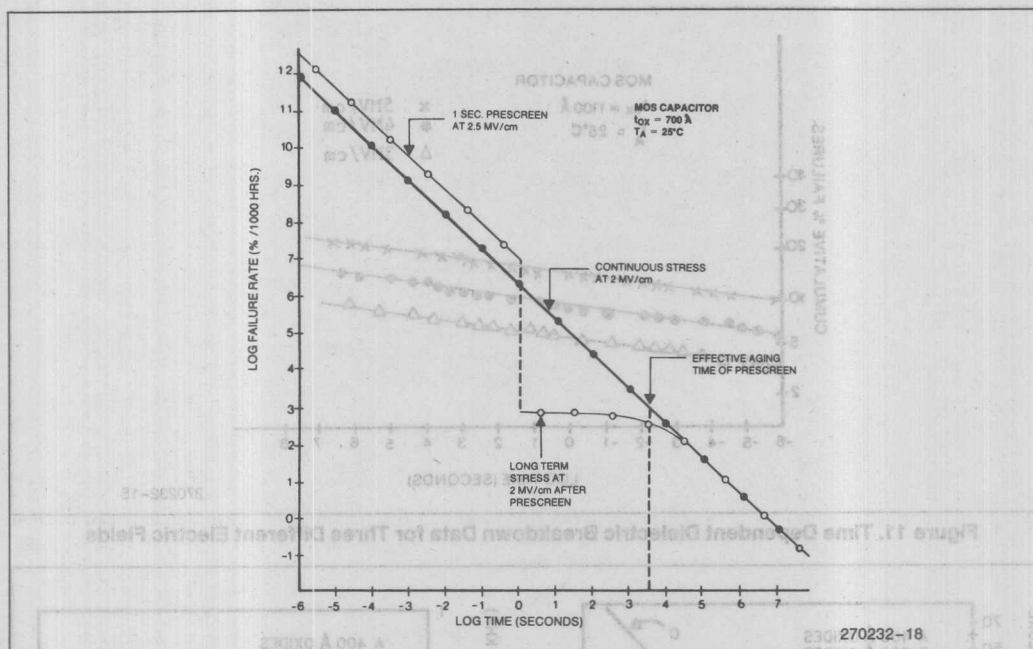


Figure 13. Instantaneous Failure Rate as a Function of Time with and without Prescreen

An oxide growth and anneal process is qualified when the latent defect density is below the target level and when there will be no oxide wearout either during normal operation or high voltage screening. Also, based on those studies, the time and voltage combination is chosen for a high-voltage node stress, which will be used to screen out virtually all devices with oxides that are defective enough to fail during the life of the part. Because the defect densities are kept very low, the fallout to the high voltage screens is generally below a few percent.

EPROM Wearout Mechanisms

EPROMs are not only subject to the above-mentioned wearout mechanisms, but are also subject to failure mechanisms which impact their data storage capability. Failure of their non-volatility can be associated with the program-erase cycling process or the data retention process. The relative impact of each of the potential failure mechanisms is a strong function of the details of the device structure and of the electric field configurations during operation. The failure mechanisms themselves are discussed in the section on EPROMs.

Reliability behavior of a new technology is characterized intrinsically on single cells and is characterized statistically and for defect density information on large arrays of cells. The data from these characterizations are then used to set cell margins and to determine reliability screening procedures.

Reliability-Related Layout Design Rules

In order to ensure that all circuit designs on a new technology have a reliability performance that is equivalent to the generic built-in reliability of the process, a set of circuit layout design rules is developed and/or approved by the Process Reliability Department. The most important reliability-related layout rules are: (1) those that impact metal and contact electromigration; (2) those that impact the hot carrier-associated problems of threshold instability and of avalanche breakdown and bipolar latch-up (which limit the amount of voltage on the chip); (3) the combinations of design rules that control CMOS latch-up; (4) the isolation spacings and channel lengths which limit chip voltages

due to punch-through; (5) those for the input and output structures which determine ESD susceptibility; and (6) those for guard ring and scribe line structures needed for contamination and moisture protection. These design rules are based on the results of a series of reliability evaluations. The validity of each of these sets of rules is verified under the combination of worst-case conditions for both critical dimensions and alignment.

Reliability Analysis of Device Structures and Materials

Device structure primarily impacts electric field configurations and current density profiles. The details of device structure are a strong function of the processes by which they are made. For example, the edge profile of a polysilicon gate is a function of polysilicon thickness and doping, edge oxidation and reoxidation thickness, and growth temperature. The edge structure has a direct impact on reliability parameters of oxide breakdown voltage for normal single poly transistors and on the DC erase and program-disturb characteristics of dual poly EPROM transistors.

Another example of a device structure effect involves the contact step coverage by metallization which is impacted by reoxidation thickness, dielectric thickness, phosphorus content and distribution (for PSG layers), contact etch conditions (wet or dry, isotropic or anisotropic) or combination (wet-dry), glass reflow, and the metal evaporator or sputterer conditions (substrate temperature, geometrical considerations, etc.). The contact step coverage (percent reduction in metal thickness) has a direct impact on the contact sidewall electromigration failure lifetime.

The choice of materials can also have a direct impact on device reliability. For example, the insertion of a barrier material such as tungsten or tantalum silicide between the metal and the silicon diffusion in contacts can prevent silicon electromigration from the contact and hence can eliminate the leakage failure mode. However, on the cautious side, refractory metals and silicide layers need to be investigated for their mechanical stability. Therefore, thin film stresses within, imparted by new layers, are measured as part of the evaluation process. Also, alloys of aluminum with copper or magnesium have been shown to improve electromigration lifetime by tying up sites along the grain boundaries; however, each needs to be investigated for its susceptibility to corrosion. A very important aluminum alloy is silicon-doped aluminum, where the silicon has been incorporated in the metal to its saturation point. This prevents alloy pitting and metal spiking of contacts due to dissolution of the silicon from the contact.

When a new structure or material is proposed, an evaluation plan and test structures are generated in order to investigate all areas of potential reliability impact. Only processes or materials which produce inherently reliable structures are permitted. The limits of physical parameters which produce satisfactory reliability are also defined from the results of these studies.

Fundamental Reliability Studies

Fundamental reliability studies are undertaken by the Process Reliability Department when the details of a failure mechanism are not well understood or are unknown. Some examples of fundamental reliability studies include sequence of events, mechanisms, and structural dependence of electrostatic breakdown and failure on CMOS inputs; mechanisms of CMOS latch-up spreading and methods for determining the originating site; layout and structural dependences of CMOS latch-up susceptibility; layout and structural dependences of charge collection on critical nodes of alpha-particle-generated minority carriers; the impact of AC or pulsed DC operation on electromigration design rules; and contact aspect ratio and other geometrical influences on contact electromigration behavior.

The purposes of these fundamental studies are to: (1) understand in detail the physics of the failure mechanisms, (2) understand circuit configuration, layout and operating dependences, (3) use understanding of reliability physics and geometry to forecast both environmental and electrical accelerating and decelerating factors and also to develop failure rate models of each mechanism, and (4) identify the key physical and/or electrical parameters and their safe limits for process control. Finally, based on the understanding derived from these fundamental studies, methodologies are developed to deal with these issues on future technologies.

Conclusion

Process reliability infers the constant development and monitoring of the many components that affect a new technology. Each of these can have an effect on product life and therefore must be understood to ensure high reliability. The problem of keeping up with these factors is only compounded by the continual shrinking of device feature size.

PRODUCT RELIABILITY

Introduction

Intel maintains a stringent and comprehensive product reliability program. Included in this program are basic device and package technology characterization, product qualification failure mode modeling, and product monitors. The purpose of this activity is to accurately predict device failure rates as a function of time and assure that each product meets and maintains stated reliability objectives. This section discusses reliability objectives, failure rate calculations, reliability philosophy – including technology reliability and reliability improvement – and typical reliability test results.

Reliability Objectives

Intel's reliability objectives can be discussed with reference to the reliability life curve shown in Figure 14. The significant portions of this curve for normal device operation are the infant mortality, early life, and random failure periods.

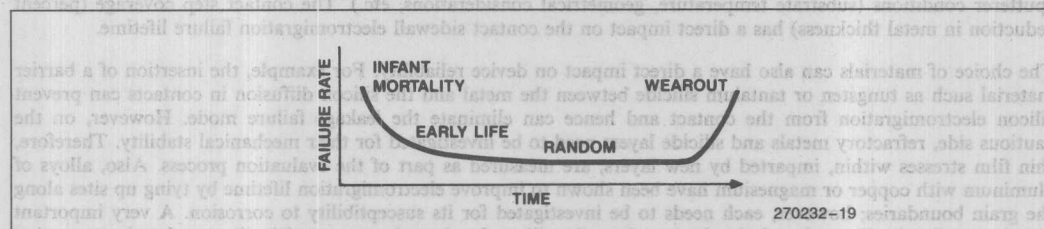


Figure 14. Reliability Life (Bathtub) Curve

Characteristic of the infant mortality and early life periods is a rapidly declining failure rate with increasing time. Subsequent to the early life period, device failure rates begin to level off at a low rate which is considered to be the random failure period. Wearout failures occur at the end of a device's useful life and are characterized by a rapidly increasing failure rate. Consistent with this discussion of the time-dependence of failure rates, Intel has defined a set of reliability objectives for each product manufactured as shown in Table 2. These objectives are more graphically illustrated with the cumulative-percent-fail-versus-time and failure-rate-versus-time graphs shown in Figures 15 and 16. Objectives are stated relative to a 55°C ambient, representing a typical worst-case "real world" application environment.

Table 2. Intel Reliability Objectives Relative to 55°C

Ambient Device Operation

Failure Period	Time Period	Failure Goal
Infant Mortality	0-300 Hours	0.05%
Early Life	0-1 Year	Avg. 200 Fit
Random	1-20 Years	100 Fit
Wearout	100,000 hours	< 0.1% Cumulative

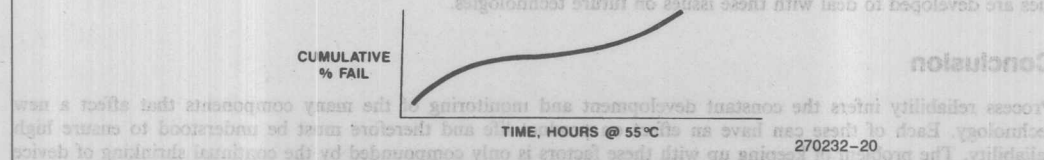


Figure 15. Cumulative Percent Failure Versus Time. Illustration of Intel's Reliability Objectives

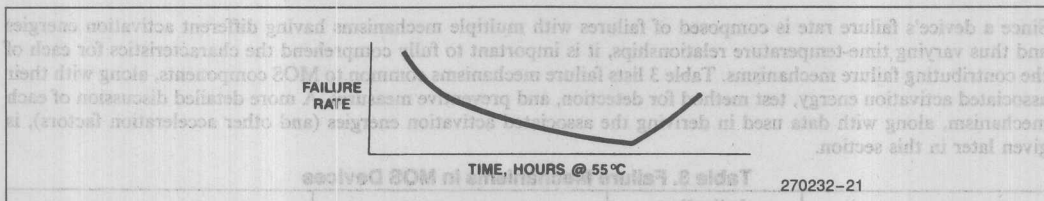


Figure 16. Failure Rate Versus Time. Illustration of Intel's Reliability Objectives

The merits of stating reliability objectives in this manner become readily apparent when determining system maintenance requirements from MTTF specifications. With an accurate representation of a component's failure rate characteristics in time, system-level reliability and maintenance needs can be more finely tuned.

Failure Rate Calculations

In determining component failure rates, it is important to fully understand the failure rate characteristics as a function of time. The most effective way to determine such characteristics is by using accelerated stress tests, either high temperature and/or high voltage. Intel's failure rate calculations are based primarily on high-temperature accelerated results where a thermal activation energy, E_a , is identified for each of the contributing failure mechanisms. Failure rate data taken at the accelerated temperature are then accurately translated to a lower temperature through the Arrhenius equation:

$$t_1/t_2 = \text{Exp} \left[\frac{E_a}{K} \left(\frac{1}{T_1} - \frac{1}{T_2} \right) \right]$$

where T_1/T_2 = temperature in °K

t_1, t_2 = MTBF at T_1 and T_2 respectively

K = Boltzmann's Constant (8.62×10^{-5} eV/°K)

E_a = Thermal Activation Energy in eV

Plots of this relationship (normalized for MTBF of one hour at 250°C) for several activation energies are shown in Figure 17.

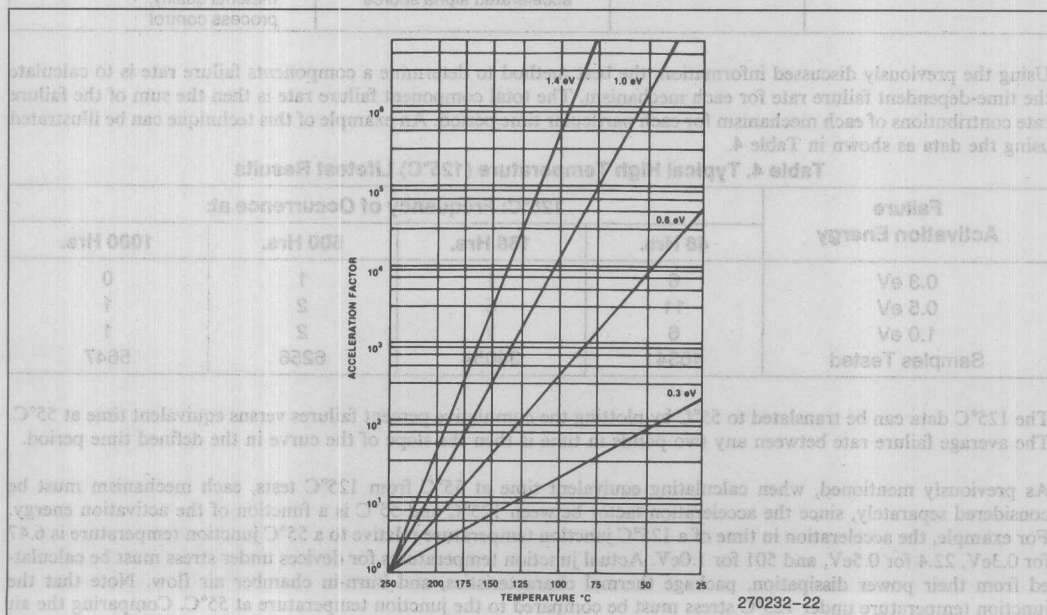


Figure 17. Arrhenius Plot

Since a device's failure rate is composed of failures with multiple mechanisms having different activation energies and thus varying time-temperature relationships, it is important to fully comprehend the characteristics for each of the contributing failure mechanisms. Table 3 lists failure mechanisms common to MOS components, along with their associated activation energy, test method for detection, and preventive measures. A more detailed discussion of each mechanism, along with data used in deriving the associated activation energies (and other acceleration factors), is given later in this section.

Table 3. Failure Mechanisms in MOS Devices

Failure Mode	Activation Energy (E_a)	Detection	Prevention
Oxide defects	0.3 eV	High voltage operating	Ultra-clean processing and high-voltage stress screens
Refresh Degradation	0.5 eV	High temp bias	Ultra-clean processing
Contamination	1.0 eV	High temp bias	Ultra-clean processing, guard rings
Silicon Defects	0.5 eV	High voltage cell stress and guardbanded tests	Quality control and ultra-clean processing and high-voltage stress screens
Metal Line Electromigration	0.5 eV	High temp operating life	Optimal design rules, process control
Contact Electromigration	0.9 eV	High temp operating life	Optimal design rules, process control
Masking defects/ Assembly defects	0.5 eV–1.0 eV	High temp storage and operating life	Quality control, inspection
Microcracks	N/A	Temp cycling	Optional design, low stress packaging, low film stresses
Short Channel Charge Trapping	–0.06 eV	Low temp, high voltage operating life	Optimal transistor design, high quality oxide, process control
Soft Error	N/A	Low voltage operation and accelerated alpha source	Optimal design, material quality, process control

Using the previously discussed information, the best method to determine a components failure rate is to calculate the time-dependent failure rate for each mechanism. The total component failure rate is then the sum of the failure rate contributions of each mechanism for each particular time period. An example of this technique can be illustrated using the data as shown in Table 4.

Table 4. Typical High Temperature (125°C) Lifetest Results

Failure Activation Energy	125°C: Frequency of Occurrence at:			
	48 Hrs.	168 Hrs.	500 Hrs.	1000 Hrs.
0.3 eV	6	1	1	0
0.5 eV	11	5	2	1
1.0 eV	6	1	2	1
Samples Tested	40341	34054	6256	5647

The 125°C data can be translated to 55°C by plotting the cumulative percent failures versus equivalent time at 55°C. The average failure rate between any two points in time is then the slope of the curve in the defined time period.

As previously mentioned, when calculating equivalent time at 55°C from 125°C tests, each mechanism must be considered separately, since the acceleration factor between 125°C and 55°C is a function of the activation energy. For example, the acceleration in time of a 125°C junction temperature relative to a 55°C junction temperature is 6.47 for 0.3eV, 22.4 for 0.5eV, and 501 for 1.0eV. Actual junction temperatures for devices under stress must be calculated from their power dissipation, package thermal characteristics, and burn-in chamber air flow. Note that the junction temperature under 125°C stress must be compared to the junction temperature at 55°C. Comparing the air temperatures, instead of the junction temperatures, is a common mistake when calculating acceleration factors.

Figure 18 shows the cumulative percent failure versus equivalent 55°C time for failures associated with each activation energy as well as the combined curve for all failures. This format is probably the most useful for presenting reliability data from a system manufacturer's standpoint, because it yields "removal rate" directly.

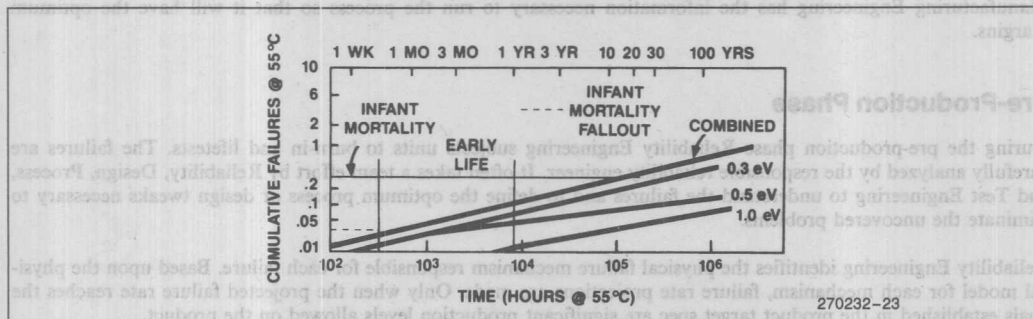


Figure 18. Cumulative Percent Failures vs. Equivalent Time at 55°C for Specific Activation Energy

Reliability Philosophy

The concept that product reliability is inherent in the technology is Intel's fundamental philosophy and is the basis for the company's reliability program. Infant mortality and early life failure rate characteristics can be improved with screening techniques – longer-term reliability is a function of the technology. Based on this premise, a key emphasis of Intel's reliability program is in basic technology and failure mechanism modeling. Intel uses, along with basic technology characterization, a lead product (typically a memory component) as the technology reliability qualification and failure mechanism characterization vehicle. Subsequent qualifications on products processed on "qualified" technologies are used to verify that no new failure mechanisms are introduced as a result of design layout error or circuit or test complexity.

The principle that product reliability is a function of technology and that products processed on the same technology will have equivalent reliability is easily illustrated by comparing static RAM and microprocessor failure rate characteristics. Figure 19 shows cumulative percent fail versus time curves of a 1K SRAM (2115H), 4K SRAM (2147H), and 8086 microprocessor, all processed on the same technology. As can be seen, except for the active area dependence, each product has equivalent reliability characteristics.

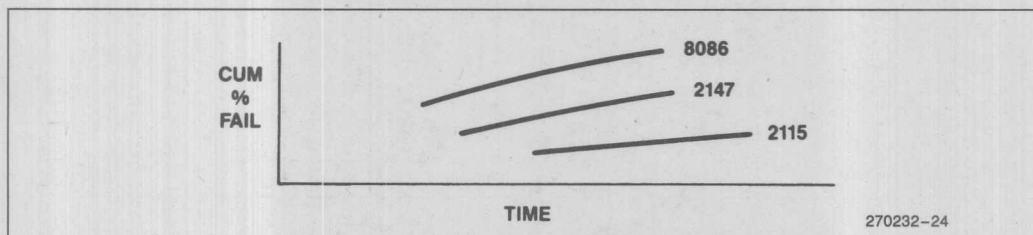


Figure 19. Failure Versus Time Characteristic Comparison of SRAMs and Microprocessors

With this emphasis on technology and failure mechanism modeling, reliability can be built into the technology and likewise into subsequent products. In addition to process and process control improvements, this understanding of the technology allows for optimization of screening techniques which further improve, in particular, the infant mortality and early life component failure rates.

Manufacturing Engineering has the information necessary to run the process so that it will have the optimum margins.

Pre-Production Phase

During the pre-production phase Reliability Engineering subjects units to burn-in and lifetests. The failures are carefully analyzed by the responsible reliability engineer. It often takes a team effort by Reliability, Design, Process, and Test Engineering to understand the failures and to define the optimum process or design tweaks necessary to eliminate the uncovered problems.

Reliability Engineering identifies the physical failure mechanism responsible for each failure. Based upon the physical model for each mechanism, failure rate projections are made. Only when the projected failure rate reaches the goals established in the product target spec are significant production levels allowed on the product.

The development of the product is performed on the manufacturing line itself, since Intel does not have a pilot manufacturing line. This reduces the possibility of new problems occurring during high-volume production, which frequently is observed during a transition from a pilot line to a final manufacturing line. The manufacturing engineers and personnel have a major involvement in fabricating the product, long before significant production volumes are reached. The process development and the design engineers work together with the manufacturing personnel until all the manufacturing bugs have been worked out.

Conclusion

The successful development of a product requires a complete understanding of quality and reliability issues throughout the entire development cycle. Quality and reliability cannot be an after-thought.

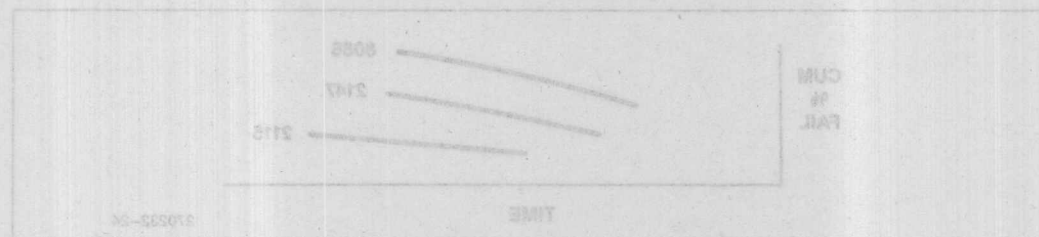


Figure 19: Failure Rate Versus Time Characteristics Comparison of SRAMs and Microprocessors

With this emphasis on technology and failure mechanism modeling, reliability can be built into the technology and likewise into subsequent products. In addition to process and process control improvements, the understanding of the technology allows for optimization of screening techniques which further improve, in particular, the infant mortality and early life component failure rates.

Reliability Program

To assure that all products meet and maintain these reliability objectives, Intel maintains a comprehensive reliability program. Within the program there are four distinct areas of focus:

- Technology Reliability
- Technology/Product Qualification
- Reliability Monitors
- Reliability Improvement

TECHNOLOGY RELIABILITY

Early in the technology/product development cycle the reliability engineering group works closely with Technology Development, Assembly Engineering and Fab Engineering to identify potential reliability concerns. Team meetings are held to specifically address these reliability concerns and establish evaluation/modeling programs. The focus of this activity is to verify design rules, such as line and contact current density rules, and model failure kinetics (time dependence and acceleration factors) for potential new concerns, such as thin dielectrics. The primary purpose is to identify limitations early so that process modifications, process control improvements, or design revisions can be made before qualification material is manufactured.

QUALIFICATION

The purpose of the Intel Automotive technology/product qualifications is to assure that the short- and long-term reliability objectives of Intel components are met. All new technologies and products or changes to existing products are subject to qualification before production shipments are made. Specific areas requiring qualification include:

1. New technology/process
2. Process modification
3. Design revision
4. Wafer fab transfer/pick-up
5. New package

In performing the qualification, a series of key tests are performed to accurately evaluate a product's failure rate characteristics and to assure that the previously discussed reliability objectives are met. These tests conclude the following:

1. High Temperature Dynamic Lifetest*
2. High Voltage Dynamic Lifetest
3. Low Temperature Dynamic Lifetest
4. High Temperature Storage (Bake)
5. High Temperature/Humidity Lifetest*
6. Temperature Cycling*

*Similar to SAE Qualification Test.

HIGH TEMPERATURE DYNAMIC LIFETEST

High temperature dynamic lifetesting is performed to measure actual field reliability. Lifetests of 1000-hour to 2000-hour duration are used to accelerate failure mechanisms by functionally operating the device at an elevated ambient temperature (125°C). Data obtained from this test are used to predict product infant mortality, early life, and random failure rates. Data are translated to standard operating temperatures by failure analysis to determine the activation energy of each of the observed failures, using the Arrhenius relationship as previously discussed. Lifetesting is performed at a 125°C ambient operated at +5 percent voltage conditions with typical or nominal timing parameters.

HIGH VOLTAGE DYNAMIC LIFETEST

High voltage dynamic lifetesting is performed to verify long-term failure rates and assure wearout objectives. Results through 1000–2000 hours are also used as an aid in identifying failure mechanisms and assessing their respective acceleration factors. Test conditions are identical to the high temperature lifetests except that voltage conditions are elevated to 7–8 volts.

LOW TEMPERATURE DYNAMIC LIFETEST

Low temperature dynamic lifetesting is used to detect the effects of hot electron injection. The conditions for hot electron injection occur during operation when the transistors are in saturation. To accelerate this effect, low temperature (-10°C), high voltage (7–8 volts) and maximum duty cycle are used. Test duration is 1000–2000 hours.

HIGH TEMPERATURE STORAGE

High temperature storage (bake) is a test in which devices are subjected to elevated temperatures with no applied bias. Temperatures used are 160°C (1000 hours) for plastic-encapsulated devices and 250°C (500 hours) for devices in hermetic packages. The test is used to detect mechanical instabilities such as bond integrity, and process wearout mechanisms such as ionic contamination, contact integrity, and in the case of EPROMs, data retention.

HIGH TEMPERATURE/HUMIDITY LIFETEST

High temperature/humidity testing is performed to evaluate moisture resistance characteristics of plastic-encapsulated components. A 2000-hour test is performed under static bias conditions at $85^{\circ}\text{C}/85$ percent relative humidity with nominal voltages. To maximize metal corrosion conditions, the biasing configuration is either under low power or no power, with alternative pins biased at +5 volts or 0 volts.

TEMPERATURE CYCLING

This test consists of cycling the temperature of a chamber housing device from -65°C to 150°C (condition C) with no applied bias. Temperature cycling (1000 cycles) is used to detect mechanical instabilities such as bond and die attach integrity as well as metal or polysilicon microcracks. Condition C is a requirement of the SAE temperature cycling qualification test.

REQUIREMENTS

Minimum sample size requirements for qualification testing vary depending on complexity. There are three categories of qualifications: new technology, process/product, revisions. Table 5 shows the types of qualifications for each category, and Table 6 shows the qualification tests and sample size requirements for each category.

Table 5. Qualification Categories

Category	Type
New Technology	<ul style="list-style-type: none"> • New Module • Process Scaling • Wafer Fab Transfer
Process/Product	<ul style="list-style-type: none"> • New Product using a Qualified Process • Package Change • Product Conversion to Qualified Process • Non Scaling Process Change
Revision	<ul style="list-style-type: none"> • Design Revision of Existing Product • Equipment Changes

Table 6. Qualification Test Requirements

Evaluation	Test	Technology Qual.	Product/ Product/Qual.	Revision Qual.
Infant Mortality	125°C Dynamic Lifetest	10K units 10 lots through 48 hours, 5K units (5 lots) through 168 hours	3-5K units 3-5 lots through 48 hours	100-300 Units 1-3 lots through 168 hours
Early Life Random	125°C Lifetest 5.0V	2000 hours, 750 units from 5 lots	1000 hours 300 units from 3 lots	1000 Hours 100-300 units from 1-3 lots
	125°C	2000 hours 500 units from 5 lots	N/A	N/A
	85/85 Plastic Only	2000 hours, 1000 units from 5 lots	1000 hours, 300 units from 3 lots	1000 hours 100-300 units from 1-3 lots
Wearout	250°C Bake EPROM	1000 hours, 250 units from 5 lots	500 hours, 150 units from 3 lots	168 hours, 100 units from 2 lots
	250°C Bake	500 hours, 125 units from 5 lots	500 hours, 75 units from 3 lots	500 hours, 25- 75 units from 1-3 lots
	-10°C Dynamic	2000 hours, 125 units from 5 lots	N/A	N/A
	Temp. Cycle	1000 cycles, 125 units from 5 lots	1000 cycles, 75 units from 3 lots	1000 cycles, 25-75 units from 1-3 lots

RELIABILITY MONITORS

Once a technology/product is qualified, a production monitor is established to assure that reliability objectives are maintained on a continuing basis. Samples from standard production material representing each technology and product family and manufacturing plant (under fab and assembly) are subjected to a series of reliability tests as defined in Table 6. Analysis is performed on all failures, and data are trended on a weekly basis. This analysis allows for rapid identification of potential problems and subsequent corrective action plans. If at any time a product's failure rate exceeds Intel's reliability objectives, a Material Review Board (MRB) is immediately formed to take the appropriate corrective actions.

RELIABILITY IMPROVEMENT

Reliability improvement is a continuing process. It starts during the initial phases of a product's development and lasts throughout its life cycle. As previously discussed, product reliability is inherent in the technology; thus much of the improvement comes from process and process control enhancements. The reliability improvement curve follows the product learning curve. That is, as the product/technology matures, yields increase primarily due to lower defect levels. Lower initial defects also yield lower latent defects, resulting in improved reliability.

In addition to technology learning, reliability learning also plays a significant role in reliability improvement. By using data and failure analysis from the qualification and monitor activity, dominant mechanisms are identified.

These mechanisms are fully characterized, resulting in optimized or new reliability screens and/or process or process control improvements. Further, since each new generation of products builds on this knowledge, there is improved reliability with each generation of products. This reliability improvement trend is illustrated by the failure rate evolution of NMOS dynamic RAMs shown in Figure 20.

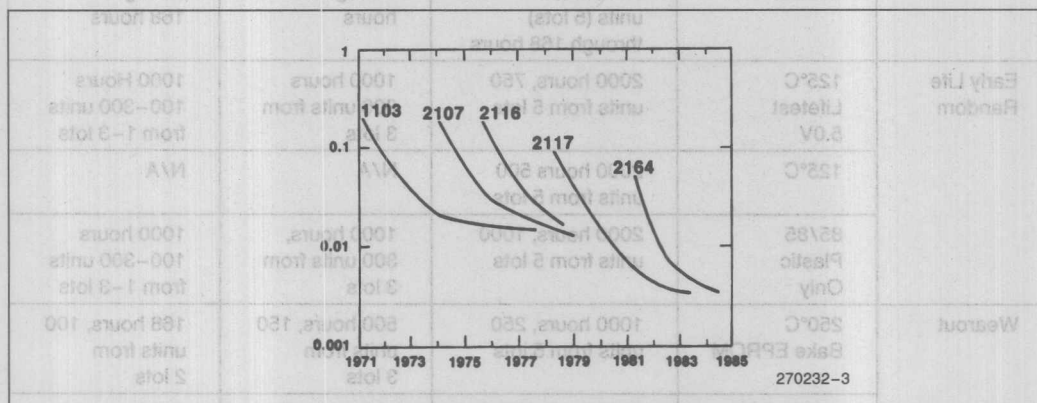


Figure 20. NMOS Dynamic RAM Failure Rate Evolution

Though long-term reliability is inherent in the technology, reliability screening techniques are effective in improving infant mortality and early life failure rates. These screens consist of thermal and/or voltage accelerated stresses to age the product, thereby achieving a lower initial failure rate. Another screening technique is the use of a guardbanded functional test to detect marginalities induced by latent defects. Figure 21 shows the effects of an aging screen on a product's failure rate characteristics. As can be seen, early failure characteristics are significantly improved, though long-term characteristics are relatively unchanged. As a result, Intel employs the use of screening techniques to improve reliability in the early stages of a product's life cycle, though the longer-term goal is to achieve reliability objectives through process and/or process control improvements.

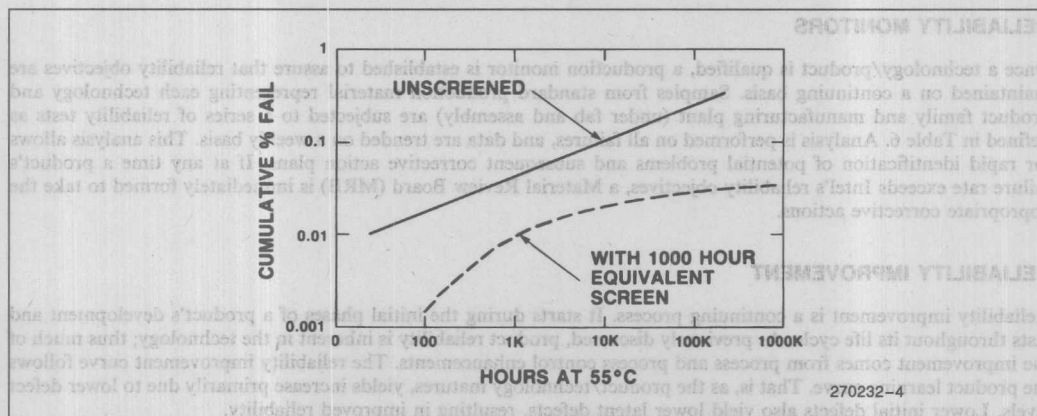


Figure 21. Effect of Aging Screen on Product Reliability

TYPICAL RELIABILITY TEST RESULTS

Table 7 shows reliability test data and failure calculations for key technology families (EPROM, SRAM, Microprocessor) based on results from Intel's Reliability Program.

Table 7. Reliability Test Data for Intel Product

Technology	No. Lots	Device Hours	Equivalent Device Hours 55°C, for $E_a = 0.3, 0.5, 1.0$ eV			Failure Rate 55°C, for $E_a = 0.3, 0.5, 1.0$ eV		
			125°C	0.3 eV	1.0 eV	0.3 eV	0.5 eV	1.0 eV
HMOS-I	409	4.29E7		2.37E8	1.35E10	70	20	0
HMOS-II	299	4.77E7		2.50E8	1.21E10	110		0
HMOS-III	12	2.33E6		1.23E7	5.92E8	80		0
HMOS-DIII	102	2.72E7		1.76E8	1.36E10	50	50	0
NMOS-5V	155	1.31E7		5.98E7	8.70E8	190	0	0
NMOS-12V	167	7.24E7		4.98E8	3.10E10	160	0	0
EPROM	213	2.01E7		1.08E8	1.29E9	50	90	20
HMOS-E	261	4.06E7		2.14E8	9.36E9	40	40	10

* $E_a = 0.6$ eV, not 0.5 eV

Conclusion

Many mechanisms affect product reliability. It is necessary to investigate each of them as thoroughly as possible in order to characterize and predict the useful life of the product. Many of these can be examined through stress tests. Other data are accumulated through monitor programs maintained on a continuing basis.

RELIABILITY MONITORS

Introduction

Reliability is designed into each component Intel manufactures. From the moment the design is put on paper, stringent reliability standards must be met at each step for a product to bear the Intel name.

Built-in reliability has been the policy of Intel ever since the first 3101 RAM was marketed in 1969. Since then, each new product Intel has introduced carries the reliability lessons learned from previous products and processes. Intel's years of leading-edge semiconductor design and manufacturing experience support the reliability of today's state-of-the-art devices.

The Intel Reliability Monitor Program, started in 1974, measures and controls reliability of Intel components. The program provides Intel with the necessary data to accurately measure and control process drifts. At present, over 60,000 components are tested each month, representing all Intel technologies, and resulting in a reliability database of over 125 million device hours. This comprehensive program establishes device acceptance, burn-in requirements, and lot acceptance criteria.

This monitor program ensures that Intel customers receive only highly reliable devices. Their products can then be designed with assurance of low down-time, lower maintenance cost, and lasting quality.

The Monitor Program

The monitor program is administered and executed by the Reliability Departments of Intel's Components Division Quality and Reliability organization. Actual testing is done by each component division's reliability department. A central monitor coordinator maintains a central database of all monitor data, standardizes procedures for all divisions, and tracks performance.

The decision on which products are to undergo the monitor program is made quarterly at each divisional manufacturing division. Typically chosen are new devices or devices recently upgraded to a new process or stepping. The program coordinator, however, ensures that a complete range of processes and products throughout the company is continually tested.

The products are selected after manufacturing testing. They are then retested under the monitor program. The devices undergo a complete 48-hour 125°C lifetest while the devices are operated to simulate normal use. The number of valid failures, i.e., those found by analysis to be due to the dynamic lifetest stress, determines the "infant mortality" failure rate.

The burn-in sample size is at least 1000 devices for each product tested. Each month, over 60,000 devices are pulled out of final test for the monitor program. This represents several million dollars per year in product alone that Intel devotes to ensuring ongoing product reliability, not to mention the countless man-hours and data processing.

After that test, approximately six percent of the devices are left on test to complete a 1000-hour lifetest at 125°C. This measures the transition between "infant mortality" and "random failure rate" in portions of the bathtub curve shown in Figure 12. A readout is taken at 168, 500 and 1000 hours to determine the time-dependent failure rate.

The resulting data on the device's failures rates are measured against reliability goals set in the target specification. Each device that fails undergoes a detailed analysis to determine the failure mechanism, its importance to the process or device, and appropriate corrective action.

Failure Analysis

The Intel Reliability Monitor Program would be value limited without one essential ingredient: failure analysis. Failures that occur at any part of the production cycle are identified, and the failure mechanism is determined using various types of analytical techniques. Included techniques are EDX/SEM, Auger or visible light. Alternately, many electrical tests are performed using bench testers, logic analyzers, oscilloscopes, etc. A newer combination of techniques employs Dynamic Fault Imaging⁹, which is a combination of voltage contrast SEM and a sophisticated image processing capability.

Failure mechanisms, once identified, lead to corrective actions and elimination of the problem. Table 8 shows the activation energies of several commonly encountered failure modes.

Table 8. Major Failure Mechanisms in a MOS and Bipolar Devices

Failure Mode	Type	Activation Energy (E _a)
Oxide Defects	Infant/Random	0.3 eV
Silicon Defects	Infant/Random	0.5 eV
Mask Defects	Random	0.5 eV
Refresh	Random	0.5 eV
Charge Loss/Gain	Infant/Random	0.6 eV
Surface Charge	Wearout	0.5–1 eV
Metal Electromigration	Wearout	0.5 eV
Polarization	Wearout	1.0 eV
Slow Trapping	Wearout	1.0 eV
Contamination	Wearout/Infant	1.0 eV
Microcracks	Random	1.0–1.4 eV
Contact Electromigration	Wearout	0.5–0.9 eV
Hot Electron Injection	Wearout	–0.06 eV

Control Limits

The several Q & R reliability groups establish the performance monitors and the appropriate control limits. The current values for burn-in/lifetest failures and for package failures are given in Table 9.

Table 9. Typical Control Limits

Mode	% Defective
Burn-In/Lifetest @125°C, 168 hr. Cum Failures	0.2
Package-Related (all stresses)	0.5

When failure rates exceed these limits, corrective actions are instituted via the Materials Review Board (MRB). These corrective actions may include testing additional samples to confirm the failure modes and extent of the problem, subjecting the new samples to more stringent tests, stopping the product flow, and adding test screens to remove defective products before shipping. Screens are kept in place until the problem has been resolved and reliability is re-established. Several techniques used by Intel to control reliability once problems are identified are shown in Figures 22 and 23.

9. J.I. Goldstein, D.E. Newbury, P. Exhlin, D. C. Joy, C. Fiori, E. Lifshin, "Scanning Electron Microscopy and X-Ray Microanalysis", (1981), Plenum Press.

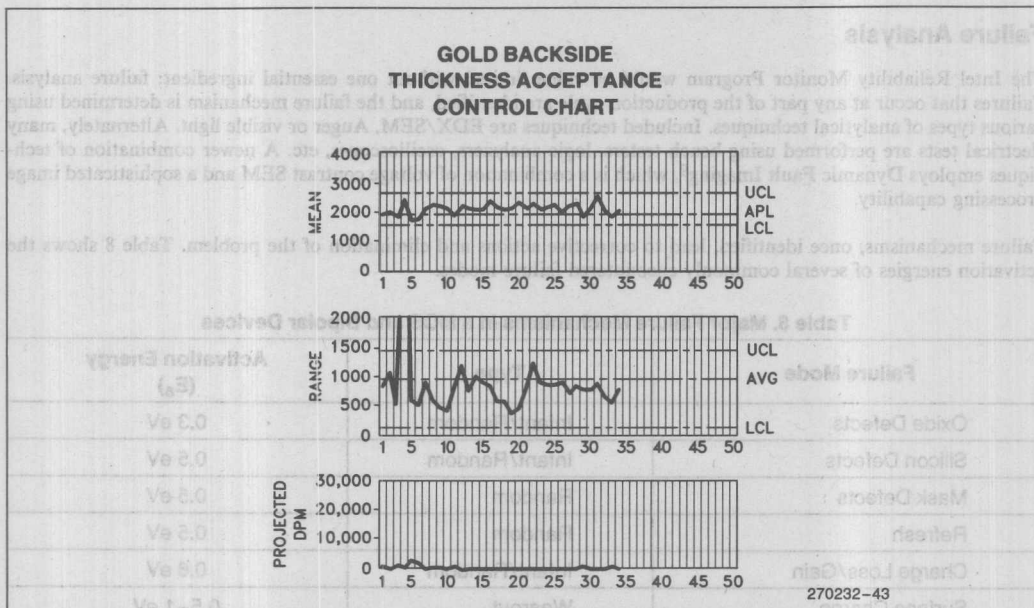


Figure 22. Typical Control Limit Chart Used in Intel Wafer Fab

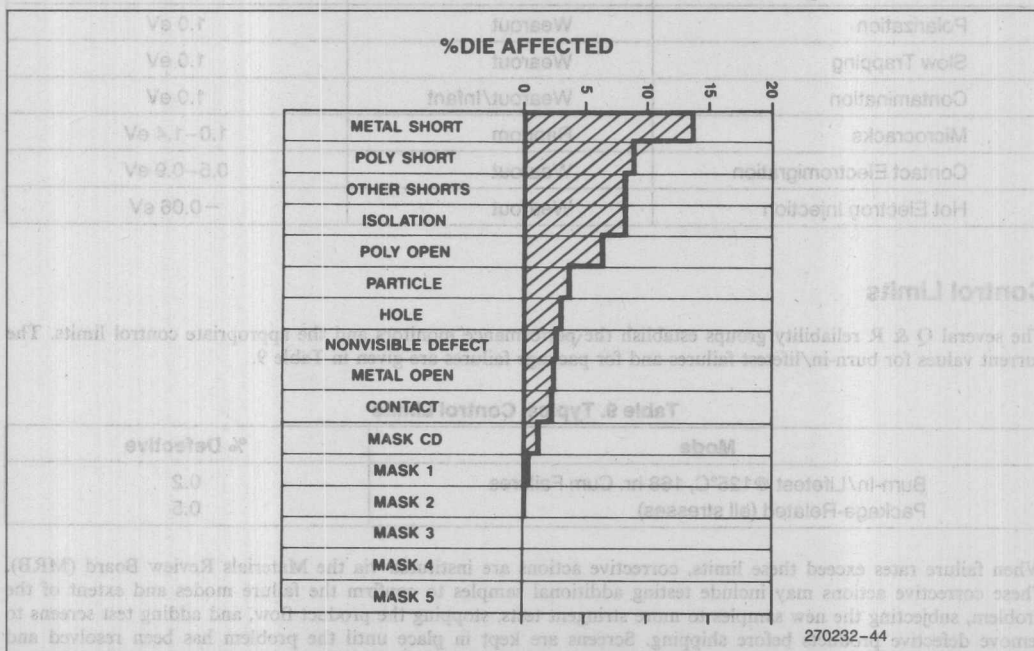


Figure 23. Pareto Analysis of Sort Visual Rejects

Summary of Reliability Results by Technology, 48 Hour Burn-In

1. Timeframe:
2. Infant Mortality—Defined as those valid failures after 48 hour dynamic burn-in at 125°C.
3. Products Sampled:

Technology	No. of Lots Sampled	Total No. of Devices	No. Failed	% Defective
4.	5.	6.	7.	8.

1. The time span over which the product was sampled.
2. The definition of 'Infant Mortality'.
3. Those products included in the sample from the process technologies.
4. The Intel technology family.
5. Number of distinct lots of product sampled during the timeframe.
6. Total numbers of devices from a technology submitted to the process monitors.
7. Number of valid failures in the timeframe for that technology.
8. Percent of failures in the timeframe for that technology.

Summary of Reliability Results any Technology, 1000 Hour Lifetest

1. Timeframe:
2. Failure rate: Defined as that rate predicted at 55°C 60% UCL, as extrapolated from 1000 hours, 125°C, dynamic lifetest data.
3. Technology:
4. Products sampled:

No. of lots sampled	Actual device hrs. at 125°C	Activation energy E _(ACT) eV	# Fail at E _(ACT)	Equiv device hrs. at 55°C	Failure rate at 55°C(60% UCL) %/1000 hrs	FITS
5.	6.	7.	8.	9.	10.	11.
*Combined failure rate.					12.	13.

1. The time span over which the product was sampled.
 2. Definition of failure rate.
 3. The Intel technology family.
 4. Those products included in the sample from the process technology.
 5. Number of distinct lots of product sampled during the timeframe.
 6. Total accumulated device hours at 125°C.
 7. Activation energy of the experimentally validated failure mode.
 8. Number of devices failed at that activation energy.
 9. Equivalent device hours at that activation energy extrapolated to 55°C.
 10. Failure rate of that failure mechanism at 55°C (60% UCL) in %/1000 hours.
 11. The same failure rate as 10., only expressed in 'FITs', failures in 10⁹ hours.
 12. Combined failure rate for that technology at 55°C (60% UCL) in %/1000 hours.
 13. The same failure rate as 12., only expressed in 'FITs', failures in 10⁹ hours.
- *(Combined failure rate does not include 48 hour burn-in data.)

Figure 24. Data Format Summary of Reliability Results by Technology

The upper and lower control limits (UCL/LCL) for these processes are shown. In addition to the control charts, Intel employs a large variety of statistical tools to analyze product reliability. Having defined the fault, the pertinent information is immediately sent to the responsible operation. For instance, if a bond pull is below spec, the relevant information is sent to Assembly Sustaining Engineering and to the site manufacturing organizations. They will revise, repair, or replace the bonding procedure to eliminate the problem, if caused by an assembly problem. However, other groups may have to be involved to solve the problem. Fab may need to fix its bond pad opening etch procedures so that residual glass does not remain on the pad. Alternately, Materials Quality (MQ) may have to modify the leadframe inspection methods to assure proper bond finger plating. Such potential fixes are generally discussed in joint MRB meetings of all concerned organizations.

The Intel Reliability Monitor Program is administered by the Intel Corporate Components Coordinator. Each operation's reliability department works closely with the Coordinator to assure that representative samples of that group's products are rapidly tested immediately upon manufacture. Reliability results categorized by product and technology are issued at periodic intervals. An explanation of the report as given in the Reliability Monitor Brochure, Order No. 210301, is shown in Figure 24.

Copies of Reliability Reports are available to customers from the Intel Literature Department. This information, part of the corporate database, is updated semiannually. Specific product summaries can also be obtained from Intel Literature Department.

Conclusion

The Intel Reliability Monitor Program tests thousands of components per month in order to detect any change in the manufacturing process that could affect reliability. Any variance from Intel standards is quickly detected, and corrective actions are taken immediately to ensure continuity of product performance, quality, and reliability.

No. of lots sampled	Actual device hrs. at 125°C	Activation energy (eV) at 125°C	Fail device hrs. at 85°C	Equip. device hrs. at 85°C (50% UCL)	Failure rate at 85°C (FITs)
2	8	7	8	8	10
11					11
12					12

*Combined failure rate

1. The time span over which the product was sampled.
 2. Definition of failure rate.
 3. The Intel technology family.
 4. Those products included in the sample from the process technology.
 5. Number of distinct lots of product sampled during the timeframe.
 6. Total accumulated device hours at 125°C.
 7. Activation energy of the experimentally validated failure mode.
 8. Number of devices failed at that activation energy.
 9. Equivalent device hours at that activation energy extrapolated to 85°C.
 10. Failure rate of that failure mechanism at 85°C (50% UCL) in 1000 hours.
 11. The same failure rate as 10, only expressed in FITs, failures in 10⁹ hours.
 12. Combined failure rate for that technology at 85°C (50% UCL) in 1000 hours.
 13. The same failure rate as 12, only expressed in FITs, failures in 10⁹ hours.
- *Combined failure rate does not include 48 hour burn-in data.

Figure 24. Data Format Summary of Reliability Results by Technology

CHAPTER 4 CUSTOMER SUPPORT

CUSTOMER FEEDBACK

INTRODUCTION

As LSI devices become more complex the testing of these devices becomes more difficult. Today testers cost upwards of \$1 million per machine. Test programs can cost an equal amount for development and continual support. Ultimately the goal of a supplier is to provide product which meets the customer's requirements and can be used with few or no incoming inspections. Programs have been implemented that have achieved this goal with several customers. However, the overall goal is not to have individual programs for each Intel customer. It is, rather, to have a total quality program in place that can accommodate all customers.

One of Intel's steps for achieving minimal or zero incoming inspection is to rely heavily on customer feedback. Customers' perception of received product quality that guarantees "data book performance" or "performance to customer specifications" is not sufficient; all product correlation issues must be resolved with the customer to gain product confidence.

Monthly, Intel monitors the quality of its performance with customers through several indicators such as percent defective at the customer's incoming product inspection, returned material, and rankings or ratings. Identifying products causing a high percentage defective at the customer's incoming inspection or the causes for a poor rating/ranking represents the start of correlative action to eliminate problems. This type of "work with" attitude creates customer confidence in using Intel's products, eventually leading to little or no incoming inspection.

CORRELATION

Many factors can cause the rejection of good units or acceptance of bad units when a customer tests Intel-supplied parts in its incoming inspection facilities. Correlation issues, in particular, can create havoc leading to line shutdown at an incoming inspection location. The following graph, Figure 1, represents the problems of guaranteeing simple correlation between customer and vendor on a given product.

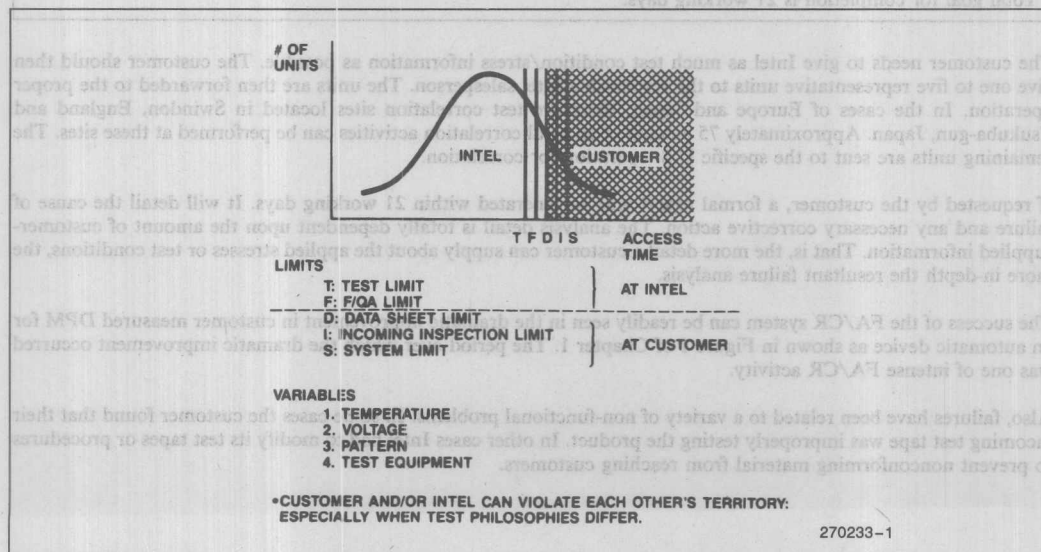


Figure 1. Access Time Correlation Example

int -1

QA group (F) will sample-test these devices at a guardbanded 247 ns. The customer may design their incoming inspection to test these devices from 250 ns to 255 ns for use in the system. (Their system should be designed to run at approximately 270 ns to 300 ns.) Variables such as testing temperatures, test patterns, test voltage, test equipment tolerances, and system design can cause measurement values to shift from 2 ns to 20 ns. If Intel and the customer do not mutually agree to guardband their measurements, there will be constant correlation issues raised between the two, where Intel passes a part that the customer may fail. In other words, Intel and its customers must agree on the test parameters so that the data distribution is acceptable to both.

FAILURE ANALYSIS/CORRELATION REPORT

However, agreement on guardbanding alone is not sufficient. Communication between Intel and the customer is truly essential. During the early 1970's Intel instituted the FA/CR (Failure Analysis/Correlation Request) system. Using this system, a customer may return failed units to Intel for failure analysis or test correlation. These failures may include qualification, incoming inspection, board/system fallout and others. The sequence of events in Table 1 shows how units can go from the customer to Intel on a FA/CR.

Table 1. FA/CR Flow

1. Customer detects a failure . . . at qualification, at incoming test or at board/system.
2. Customer/Intel Field Sales Engineer (FSE) generates a FA/CR.
3. FSE sends units to engineering support location (manufacturing site of device).
4. Failure Analysis Administrator (FAA) assigns FA/CR number for tracking.
5. FAA acknowledges receipt of FA/CR to customer and obtains necessary information for failure analysis.
6. Product Engineering performs Go/No-Go test and reports results to FAA.
7. FAA makes call to FSE or customer and reports results. If failure analysis is required, FAA directs FA/CR to appropriate engineering group.
8. Appropriate engineering group completes evaluation and generates Engineering Evaluation Report (EER) for signatures.
9. FSE communicates FA/CR results to customer.

Total goal for completion is 21 working days.

The customer needs to give Intel as much test condition/stress information as possible. The customer should then give one to five representative units to the responsible Intel salesperson. The units are then forwarded to the proper operation. In the cases of Europe and Japan, there are test correlation sites located in Swindon, England and Tsukuba-gun, Japan. Approximately 75 percent of all local correlation activities can be performed at these sites. The remaining units are sent to the specific U.S. operation for correlation.

If requested by the customer, a formal report will be generated within 21 working days. It will detail the cause of failure and any necessary corrective action. The analysis detail is totally dependent upon the amount of customer-supplied information. That is, the more detail a customer can supply about the applied stresses or test conditions, the more in-depth the resultant failure analysis.

The success of the FA/CR system can be readily seen in the dramatic improvement in customer measured DPM for an automatic device as shown in Figure 1 of Chapter 1. The period over which the dramatic improvement occurred was one of intense FA/CR activity.

Also, failures have been related to a variety of non-functional problems. In some cases the customer found that their incoming test tape was improperly testing the product. In other cases Intel had to modify its test tapes or procedures to prevent nonconforming material from reaching customers.

RETURNED MATERIAL

The FA/CR system is essential in resolving technical issues arising from testing VLSI product. In addition to the FA/CR system, Intel also analyzes the causes for material returned to the factory. Within Intel, these returns are called RMA or Returned Material Authorizations. These RMAs are analyzed to determine the return cause. If necessary, an FA/CR is generated to determine and correct the problem.

CONCLUSION

Customer confidence in Intel product is essential if minimal or no incoming inspection is to be achieved. This confidence is promoted through a formal program that regularly feeds back to Intel information on defects, returned material and other information. The serious interaction between Intel and the customer can then effectively eliminate any problems and increase the perception of high product quality and reliability.

1. M.H. Woods, S. Rosenberg, "EPROM Reliability", *Electronics* 53, (1980), p. 133.
2. D.L. Crook, "Method of Determining Reliability Scores for Time Dependent Detection", *Proceedings International Reliability Physics Symposium* 7, (1979), p. 1.
3. E. Philofsky, "Purple Plague Revisited", *Proceedings International Reliability Physics Symposium* 12, (1979), p. 131.
4. E. Philofsky, "Intermetallic Formation in Gold-Aluminum Systems", *Proceedings International Reliability Physics Symposium* 19, (1981), p. 182.
5. G.G. Hartman, "Metallurgical Failure Modes of Wire Bond", *Proceedings International Reliability Physics Symposium* 9, (1974), p. 114.
6. R.E. Shinn, J.M. Gaywood, H.L. Euzant, "Data Retention in EPROMs", *Proceedings International Reliability Physics Symposium* 18, (1980), p. 238.
7. N. Mielke, "New EPROM Data Loss Mechanisms", *Proceedings International Reliability Physics Symposium* 21, (1983), p. 106.
8. B. Unger, "Electrostatic Discharge Failure Mechanisms and Models", *Semiconductor International* 3, (1982), p. 197.
9. R. Shinn, R. Hap, N. Mielke, "Characterization and Screening of 5K Defects in EPROM Structures", *Proceedings International Reliability Physics Symposium* 21, (1983), p. 248.
10. I.A. Blech, E. S. Meisner, "Electromigration in Thin Al Films", *Journal of Applied Physics* 40, (1969), p. 482.
11. S. Vaidya and A. K. Sinha, "Electromigration Induced Leakage at Shallow Junction Contacts Metallized with Aluminum-Poly-Silicon", *Proceedings International Reliability Physics Symposium* 20, (1982), p. 50.
12. I.A. Blech, "Copper Electromigration in Aluminum", *Journal Applied Physics* 48, (1977), p. 473.
13. C. Hu, "Hot Electron Effects in MOSFETs", *IDEM Tech. Digest, International Electron Device Meeting*, Wash. D.C. (1983), p. 176.
14. S. Timoshenko, J. N. Goodier, "Theory of Elasticity", (1951), p. 78, McGraw-Hill.
15. Acceleration Factors for Thin Gate Oxide Gussing, J.W. McPherson, D.A. Bagley, 1982 IRPS Proceedings, pp. 1-3.
16. A Non-Aging Screen to Prevent Wearout of Ultra-Thin Dielectrics, W. Meyer, D. Crook, 1983 IRPS Proceedings, pp. 6-10.
17. A.D. Smigelskas, E. O. Kirkendall, *Trans AIME* 177, (1947), p. 130.
18. F. Seitz, "On the Porosity Observed in the Kirkendall Effect", *Acta Met* 1, (1953), p. 352.
19. I.A. Blech, H. Sello, "Some New Aspects of Gold-Aluminum Bonds", *Journal Electrochem. Cal. Society* 113, (1966), p. 1032.
20. R.E. Thomas, V. Winchell, K. Jones, T. Schurr, "Plastic Outgassing Induced Wire Bond Failure", *Proceedings Electronic Electromagnetic Components Conference* 27, (1977), p. 182.
21. R.C. Bligh II, I. Farook, "Wire Bond Integrity Test Chip", *Proceedings International Reliability Physics Symposium* 21, (1983), p. 142.
22. P.H. Van Lee, *Acta Met* 10, (1962), p. 1089.
23. S.U. Campbell, E. Fott, G. Rimini, S. Lau, J.W. Major, *Phil Magazine* 11, (1974), p. 92.
24. "Oxidation Potentials of the Elements", *Handbook of Chemistry and Physics*, (1958), p. 1933, Chemical Rubber Co., Publishers.
25. N.L. Soter, R.P. Kozminski, "New Accelerated Factors for Temperature Humidity Bias Testing", *Proceedings International Reliability Physics Symposium* 16, (1978), p. 151.
26. D.S. Fock, "Analysis of Data from Accelerated Stress Tests", *Proceedings International Reliability Physics Symposium* 9, (1976), p. 143.

BIBLIOGRAPHY

1. T.C. May, M. Woods, "A New Physical Model for Soft Errors in Dynamic Memories", Proceedings International Reliability Physics Symposium 16, (1978), p. 33.
2. P.A. Gargini, C. Tseng, M. H. Woods, "Elimination of Silicon Electromigration in Contacts by the Use of an Interposed Barrier Metal", Proceedings International Reliability Physics Symposium 20, (1982), p. 66.
3. M.H. Woods, S. Rosenberg, "EPROM Reliability", Electronics 53, (1980), p. 132.
4. D.L. Crook, "Method of Determining Reliability Screens for Time Dependent Dielectric Breakdown", Proceedings International Reliability Physics Symposium 17, (1979), p. 1.
5. E.S. Meieran, P.R. Engel, T.C. May, "Measurement of Alpha Particle Radioactivity in I.C. Device Packages", Proceedings International Reliability Physics Symposium, 17, (1979), p. 13.
6. C. Quate, J. Calhoun, "Film Adhesion Studies with the Acoustic Microscope", Thin Solid Films 74, (1980), p. 295.
7. E. Philofsky, "Purple Plague Revisited", Proceedings International Reliability Physics Symposium 12, (1974), p. 131.
8. E. Philofsky, "Intermetallic Formation in Gold-Aluminum Systems", Proceedings International Reliability Physics Symposium 19, (1981), p. 182.
9. G.G. Harman, "Metallurgical Failure Modes of Wire Bond", Proceedings International Reliability Physics Symposium 9, (1974), p. 114.
10. R.E. Shiner, J.M. Caywood, B.L. Euzent, "Data Retention in EPROMs", Proceedings International Reliability Physics Symposium 18, (1980), p. 238.
11. N. Mielke, "New EPROM Data Loss Mechanisms", Proceedings International Reliability Physics Symposium 21, (1983), p. 106.
12. B. Unger, "Electrostatic Discharge Failure Mechanisms and Models", Semiconductor International 5, (1982), p. 197.
13. R. Shiner, R. Haq, N. Mielke, "Characterization and Screening of SiO Defects in EPROM Structures", Proceedings International Reliability Physics Symposium 21, (1983), p. 248.
14. I.A. Blech, E. S. Meieran, "Electromigration in Thin Al Films", Journal of Applied Physics 40, (1969), p. 485.
15. S. Vaidya and A. K. Sinha, "Electromigration Induced Leakage at Shallow Junction Contacts Metallized with Aluminum/Poly-Silicon", Proceedings International Reliability Physics Symposium, 20 (1982), p. 50.
16. I.A. Blech, "Copper Electromigration in Aluminum", Journal Applied Physics 48, (1977), p. 473.
17. C. Hu, "Hot Electron Effects in MOSFETS", IDEM Tech. Digest, International Electron Device Meeting, Wash. D.C. (1983), p. 176.
18. S. Timoshenko, J. N. Goodier, "Theory of Elasticity", (1951), p. 78, McGraw-Hill.
- 18A. Acceleration Factors for Thin Gate Oxide Stressing, J.W. McPherson, D.A. Bagler, 1985 IRPS Proceedings, pp. 1-5.
- 18B. A Non-Aging Screen to Prevent Wearout of Ultra-Thin Dielectrics, W. Meyer, D. Crooks, 1985 IRPS Proceedings, pp. 6-10.
19. A.D. Smigelskas, E. O. Kirkendall, Trans AIME 171, (1947), p. 130.
20. F. Seitz, "On the Porosity Observed in the Kirkendall Effect", Acta Met 1, (1953), p. 355.
21. I.A. Blech, H. Sello, "Some New Aspects of Gold Aluminum Bonds", Journal Electrochem, Cal. Society 113, (1966), p. 1052.
22. R.E. Thomas, V. Winchell, K. Jones, T. Scharr, "Plastic Outgassing Induced Wire Bond Failure", Proceedings Electronic Electromagnetic Components Conference 27, (1977), p. 182.
23. R.C. Blish II, L. Parobek, "Wire Bond Integrity Test Chip", Proceedings International Reliability Physics Symposium 21, (1983), p. 142.
24. P.H. Van Lent, Acta Met 10, (1962), p. 1089.
25. S.U. Campisano, L. Foti, G. Rimini, S. Lau, J.W. Major, Phil Magazine 31, (1974), p. 95.
26. "Oxidation Potentials of the Elements", Handbook of Chemistry and Physics, (1958), p. 1733, Chemical Rubber Co., Publishers.
27. N.L. Sbar, R.P. Kozadiewicz, "New Accelerated Factors for Temperature, Humidity, Bias Testing", Proceedings International Reliability Physics Symposium 16, (1978), p. 161.
28. D.S. Peck, "Analysis of Data from Accelerated Stress Tests", Proceedings International Reliability Physics Symposium 9, (1976), p. 143.

29. J.W. Peeples, "Influence of Electrical Bias Level on 85/81 test Results of Plastic Encapsulated 4K RAMs", *Proceedings International Reliability Physics Symposium* 16, (1978), p. 154.
30. G.L. Schnable, R.B. Comizzoli, W. Kern and L.K. White, "A Survey of Corrosion Failure Mechanisms in Microelectronic Devices", *RCA Review*, 40, (1979), p. 416.
31. P. Dumoulin, J.P. Seurin, P. Marce, "Metal Migrations Outside During Accelerated Life Tests", *Proceedings of Electronic Comp. Conference* 32, (1982), p. 229.
32. J. Kiely, "Simulating the Corrosion Threshold of LSI/VLSI Devices Using Moisture Sensitive Test Patterns", *Moisture Measurement and Control of Semi-Conductor Devices III*, (1983), RADC/NBS Workshop.
33. J. Kiely, personal communication.
34. *Metals Handbook* 1, 8th Ed, (1964), p. 276.
35. *Metals Handbook* 1, 8th Ed, (1964), p. 232.
36. E.S. Meieran, T.R. Cass, "The Application of Scanning and Transmission Electron Microscopy in the Semiconductor Industry", *Electron Microscopy and Structure Materials*, University of California Press, Berkeley, Ca. (1971), p. 1027.
37. E.S. Meieran, D.L. Crosthwait, J.R. Devaney, "SEM Techniques for Semiconductor Device Studies", *Scanning Electron Microscopy*, (1975), p. 715.
38. J.W. S. Hearle, J.T. Sparrow, P.M. Cross, "The Use of the Scanning Electron Microscope", (1972), Pergamon Press.
39. D.B. Holt, M.D. Muir, P.R. Grant, I.M. Boswarva, "Quantitative Scanning Electron Microscopy", (1974), Academic Press.
40. J.I. Goldstein, D.E. Newbury, P. Exhlin, D.C. Joy, C. Fiori, E. Lifshin, "Scanning Electron Microscopy and X-Ray Microanalysis", (1981), Plenum Press.
41. A.W. Agar, R.H. Alderson, D. Chescoe, "Principles and Practice of Electron Microscope Operation", (1974), North-Holland.
42. C.J. Powell, "The Physical Basis for Quantitative Surface Analysis by Auger Electron Spectroscopy and X-Ray Photoelectron Spectroscopy", in "Quantitative Surface Analysis of Materials", N.W. McIntyre, ed., ASTM Special Publication 643, ASTM, (1978).
43. R.K. Lowry and A.W. Hogrefe, "Applications of Auger and Photoelectron Spectroscopy in Characterizing IC Materials", *Solid State Technology* 23, (1980), p. 171.
44. K. Ura, H. Fujioka, K. Nakamae, M. Ishisaka, "Stroboscopic Observation of Passivated Microprocessor Chips by Scanning Electron Microscopy", *Scanning Electron Microscopy* (1982), p. 1061.
45. A.R. Stivers, "Voltage Contrast: A Powerful Tool For VLSI Circuit Diagnosis", *Proceedings 41st Annual EMSA Meeting*, G.W. Buitey, ed., (1983), p. 164.
46. T.C. May, G.L. Scott, E.S. Meieran, P. Winer, B. Rao, "Dynamic Fault Imaging of VLSI Random Logic Devices", *Proceedings International Reliability Physics Symposium* 23, (1984).
47. K. Krishnan, J.R. Ferrero, "Techniques Used in Fourier Transform Infrared Spectroscopy", *Fourier Transform Infrared Spectroscopy* 3, J.R. Ferraro, L.J. Basile, ed., (1982), p. 149, Academic Press.
48. J.A. de Haseth, "Fourier Transform Infrared Spectrometry", *Fourier, Hadamard, and Hilbert Transforms in Chemistry*, A.G. Marshall, ed., (1982), p. 387, Plenum Press.
49. E.J. McInerney, P. A. Flinn, "Diffusivity of Moisture in Thin Films", *Proceedings International Reliability Physics Symposium* 20, (1982), p. 265.
50. M. Herman, "Theory and Application of RBS", *Intel Technical Journal* (to be published).
51. "Secondary Ion Mass Spectrometry SIMS IV", A. Benninghoven, J. Okano, R. Shimizu, H.W. Werner, eds., (1984), Springer-Verlag.
52. P.F. Schmidt, C.W. Pearce, "A Neutron Activation Analysis Study of the Sources of Transition Group Metal Contamination in the Silicon Device Manufacturing Process", *Journal Electrochemical Society* 128, (1981), p. 630.
53. E.S. Meieran, "The Application of X-Ray Topographical Techniques to the Study of Semiconductor Crystals and Devices", *Siemens Review* 37, (1970), p. 1.
54. B.K. Tanner, "X-Ray Diffraction Topography", Pergamon Press, U.K. (1976).
55. G.Z. Borrmann, *Z. Physics* 42, (1941), p. 157.
56. A.R. Lang, "Diffraction and Imaging Techniques and Material Science", North-Holland Publishing Company 623, (1978).
57. W. Nelson, "Applied Life Data Analysis", Wiley-Interscience, U.S.A., (1982).



— — — — — **D** — — — — —

CHAPTER 5 INTEL MEMORY TECHNOLOGIES

This chapter is devoted to techniques and information to help you design and implement semiconductor memory in your application or system. In this section, however, the memory chip itself will be examined and the processing technology required to turn a bare slice of silicon into high performance memory devices is described. The discussion has been limited to the basics of MOS (Metal Oxide Semiconductor) technologies as they are responsible for the overwhelming majority of memory devices manufactured at Intel.

There are three major MOS technology families—PMOS, NMOS, and CMOS (Figure 1). They refer to the channel type of the MOS transistors made with the technology. PMOS technologies implement p-channel transistors by diffusing p-type dopants (usually Boron) into an n-type silicon substrate to form the source and drain. P-channel is so named because the channel is comprised of positively charged carriers. NMOS technologies are similar, but use n-type dopants (normally phosphorus or arsenic) to make n-channel transistors in p-type silicon substrates. N-channel is so named because the channel is comprised of negatively charged carriers. CMOS or Complementary MOS technologies combine both p-channel and n-channel devices on the same silicon. Either p- or n-type silicon substrates

can be used, however, deep areas of the opposite doping type (called wells) must be defined to allow fabrication of the complementary transistor type.

Most of the early semiconductor memory devices, like Intel's pioneering 1103 dynamic RAM and 1702 EPROM were made with PMOS technologies. As higher speeds and greater densities were needed, most new devices were implemented with NMOS. This was due to the inherently higher speed of n-channel charge carriers (electrons) in silicon along with improved process margins. The majority of MOS memory devices in production today are fabricated with NMOS technologies. CMOS technology has begun to see widespread commercial use in memory devices. It allows for very low power devices and these have been used for battery operated or battery back-up applications. Historically, CMOS has been slower than any NMOS device. Recently, however, CMOS technology has been improved to produce higher speed devices. Up to now, the extra cost processing required to make both transistor types has kept CMOS memories limited to those areas where the technology's special characteristics would justify the extra cost. In the future, the learning curve for high performance CMOS costs will make a larger and larger number of memory devices practical in CMOS.

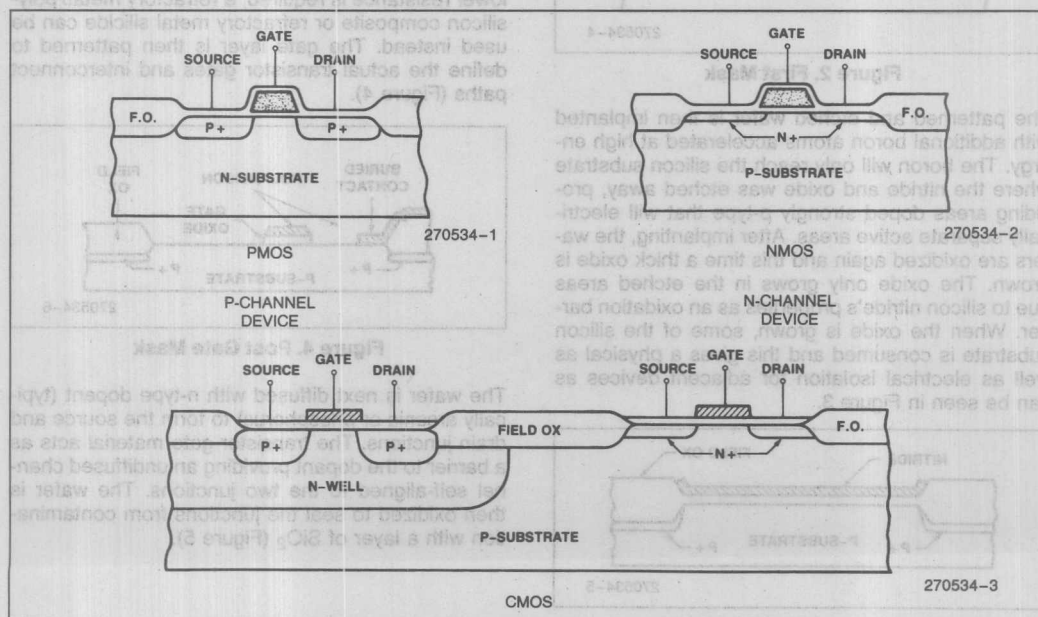


Figure 1. MOS Process Cross-sections

In the following section, the basic fabrication sequence for an HMOS circuit will be described. HMOS is a high performance n-channel MOS process developed by Intel for 5V single supply circuits. HMOS, along with its evolutionary counterparts HMOS II and HMOS III, CHMOS and CHMOS II (and their variants), comprise the process family responsible for most of the memory components produced by Intel today.

The MOS IC fabrication process begins with a slice (or wafer) of single crystal silicon. Typically, it's 100 or 150 millimeter in diameter, about a half millimeter thick, and uniformly doped p-type. The wafer is then oxidized in a furnace at around 1000°C to grow a thin layer of silicon dioxide (SiO_2) on the surface. Silicon nitride is then deposited on the oxidized wafer in a gas phase chemical reactor. The wafer is now ready to receive the first pattern of what is to become a many layered complex circuit. The pattern is etched into the silicon nitride using a process known as photolithography, which will be described in a later section. This first pattern (Figure 2) defines the boundaries of the active regions of the IC, where transistors, capacitors, diffused resistors, and first level interconnects will be made.

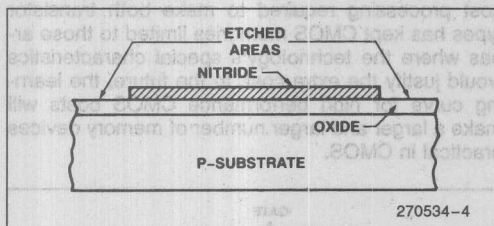


Figure 2. First Mask

The patterned and etched wafer is then implanted with additional boron atoms accelerated at high energy. The boron will only reach the silicon substrate where the nitride and oxide was etched away, providing areas doped strongly p-type that will electrically separate active areas. After implanting, the wafers are oxidized again and this time a thick oxide is grown. The oxide only grows in the etched areas due to silicon nitride's properties as an oxidation barrier. When the oxide is grown, some of the silicon substrate is consumed and this gives a physical as well as electrical isolation for adjacent devices as can be seen in Figure 3.

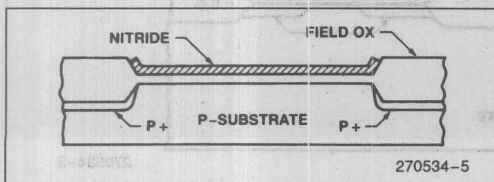


Figure 3. Post Field Oxidation

Having fulfilled its purpose, the remaining silicon nitride layer is removed. A light oxide etch follows taking with it the underlying first oxide but leaving the thick (field) oxide.

Now that the areas for active transistors have been defined and isolated, the transistor types needed can be determined. The wafer is again patterned and then if special characteristics (such as depletion mode operation) are required, it is implanted with dopant atoms. The energy and dose at which the dopant atoms are implanted determines much of the transistor's characteristics. The type of the dopant provides for depletion mode (n-type) or enhancement mode (p-type) operation.

The transistor types defined, the gate oxide of the active transistors are grown in a high temperature furnace. Special care must be taken to prevent contamination or inclusion of defects in the oxide and to ensure uniform consistent thickness. This is important to provide precise, reliable device characteristics. The gate oxide layer is then masked and holes are etched to provide for direct gate to diffusion ("buried") contacts where needed.

The wafers are now deposited with a layer of gate material. This is typically poly crystalline silicon ("poly") which is deposited in a gas phase chemical reactor similar to that used for silicon nitride. The poly is then doped (usually with phosphorus) to bring the sheet resistance down to 10-20 Ω /square. This layer is also used for circuit interconnects and if a lower resistance is required, a refractory metal/poly-silicon composite or refractory metal silicide can be used instead. The gate layer is then patterned to define the actual transistor gates and interconnect paths (Figure 4).

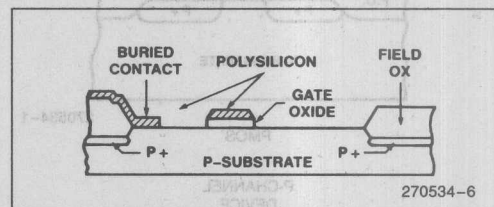


Figure 4. Post Gate Mask

The wafer is next diffused with n-type dopant (typically arsenic or phosphorus) to form the source and drain junctions. The transistor gate material acts as a barrier to the dopant providing an undiffused channel self-aligned to the two junctions. The wafer is then oxidized to seal the junctions from contamination with a layer of SiO_2 (Figure 5).

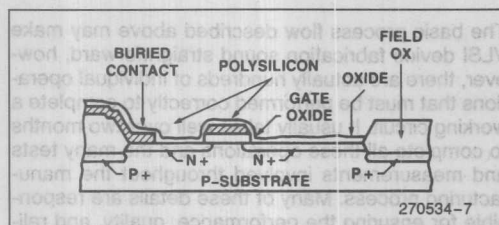


Figure 5. Post Oxidation

A thick layer glass is then deposited over the wafer to provide for insulation and sufficiently low capacitance between the underlying layers and the metal interconnect signals. (The lower the capacitance, the higher the inherent speed of the device.) The glass layer is then patterned with contact holes and placed in a high temperature furnace. This furnace step smooths the glass surface and rounds the contact edges to provide uniform metal coverage. Metal (usually aluminum or aluminum/silicon) is then deposited on the wafer and the interconnect patterns and external bonding pads are defined and etched (Figure 6). The wafers then receive a low temperature (approximately 500°C) alloy that insures good ohmic contact between the Al and diffusion or poly.

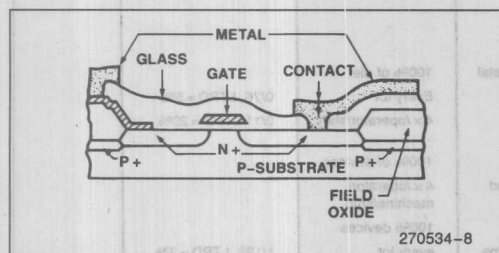


Figure 6. Complete Circuit (without passivation)

At this point the circuit is fully operational, however, the top metal layer is very soft and easily damaged by handling. The device is also susceptible to contamination or attack from moisture. To prevent this the wafers are sealed with a passivation layer of silicon nitride or a silicon and phosphorus oxide composite. Patterning is done for the last time opening up windows only over the bond pads where external connections will be made.

This completes basic fabrication sequence for a single poly layer process. Double poly processes such as those used for high density Dynamic RAMs, EPROMs, and E²PROMs follow the same general process flow with the addition of gate, poly deposition, doping, and interlayer dielectric process modules required for the additional poly layer (Figure 7). These steps are performed right after the active areas have been defined (Figure 3) providing the capacitor or floating gate storage nodes on those devices.

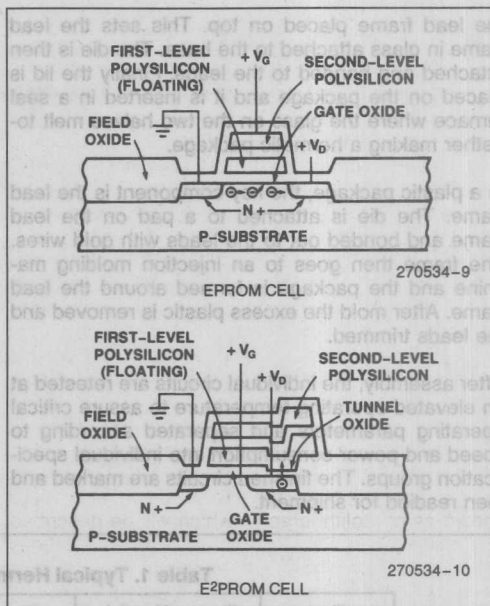


Figure 7. Double Poly Structure

After fabrication is complete, the wafers are sent for testing. Each circuit is tested individually under conditions designed to determine which circuits will operate properly both at low temperature and at conditions found in actual operation. Circuits that fail these tests are inked to distinguish them from good circuits. From here the wafers are sent from assembly where they are sawed into individual circuits with a paper-thin diamond blade. The inked circuits are then separated out and the good circuits are sent on for packaging.

Packages fall into two categories—hermetic and non-hermetic. Hermetic packages are Cerdip, where two ceramic halves are sealed with a glass frit, or ceramic with soldered metal lids. An example of hermetic package assembly is shown in Table 1. Non-hermetic packages are molded plastics.

The ceramic package has two parts, the base, which has the leads and die (or circuit) cavity, and the metal lid. The base is placed on a heater block and a metal alloy preform is inserted. The die is placed on top of the preform which bonds it to the package. Once attached, wires are bonded to the circuit and then connected to the leads. Finally the package is placed in a dry inert atmosphere and the lid is soldered on.

The cerdip package consists of a base, lead frame, and lid. The base is placed on a heater block and

frame of glass attached to the base. The die is then attached and bonded to the leads. Finally the lid is placed on the package and it is inserted in a seal furnace where the glass on the two halves melt together making a hermetic package.

In a plastic package, the key component is the lead frame. The die is attached to a pad on the lead frame and bonded out to the leads with gold wires. The frame then goes to an injection molding machine and the package is formed around the lead frame. After mold the excess plastic is removed and the leads trimmed.

After assembly, the individual circuits are retested at an elevated operating temperature to assure critical operating parameters and separated according to speed and power consumption into individual specification groups. The finished circuits are marked and then readied for shipment.

VLSI device fabrication sounds straightforward, however, there are actually hundreds of individual operations that must be performed correctly to complete a working circuit. It usually takes well over two months to complete all these operations and the many tests and measurements involved throughout the manufacturing process. Many of these details are responsible for ensuring the performance, quality, and reliability you expect from Intel products. The following sections will discuss the technology underlying each of the major process elements mentioned in the basic process flow.

PHOTOLITHOGRAPHY

The photo or masking technology is the most important part of the manufacturing flow if for no other

Table 1. Typical Hermetic Package Assembly

Flow	Process/Materials	Typical Item	Frequency	Criteria
	Wafer			
	Die saw, wafer break			
	Die wash and plate			
	Die visual inspection	Passivation, metal	100% of die	
	QA gate		Every lot	0/76, LTPD = 5%
	Die attach (Process monitor)	Wet out	4 x /operator/shift	0/11 LTPD = 20%
	Post die attach visual		100% of devices	
	Wire bond (Process monitor)	Orientation, lead dressing, etc.	4 x /operator/machine/shift	
	Post bond inspection		100% devices	
	QA gate	All previous items	every lot	1/129, LTPD = 3%
	Seal and Mark (Process monitor)	Cap align, glass integrity, moisture	4 x /furnace/shift	0/15, LTPD = 15%
	Temp cycle		10 x to mil std, 883 cond. C	1/11, LTPD = 20%
	Hermeticity check (Process monitor)	F/G leak	100% devices	
	Lead Trim (Process monitor)	Burrs, etc. (visual) Fine leak	4 x /station/shift 2 x /station/shift	0/15, LTPD = 15% 1/129, LTPD = 3%
	External visual	Solder voids, cap alignment, etc.	100% devices	
	QA gate	All previous items	All lots	1/129, LTPD = 3%
	Class test (Process monitor)	Run standards (good and reject) Calibrate every system using "autover" program	Every 48 hrs.	
	Mark and Pack			
2.	Final QA	(See attached)		

NOTES:

1. Units for assembly reliability monitor.
2. Units for product reliability monitor.

270534-11

reason than the number of times it is applied to each wafer. The manufacturing process gets more complex in order to make smaller and higher performance circuits. As this happens the number of masking steps increases, the features get smaller, and the tolerance required becomes tighter. This is largely because the minimum size of individual pattern elements determine the size of the whole circuit, effecting its cost and limiting its potential complexity. Early MOS IC's used minimum geometries (lines or spaces) of 8–10 microns (1 micron = 10^{-6} meter \approx 1/25,000 inch). The n-channel processes of the mid 1970's brought this down to approximately 5 microns, and today minimum geometries are less than 2 microns in production. This dramatic reduction in feature size was achieved using the newer high resolution photo resists and optimizing their processing to match improved optical printing systems.

A second major factor in determining the size of the circuit is the registration or overlay error. This is how accurately one pattern can be aligned to a previous one. Design rules require that space be left in all directions according to the overlay error so that unrelated patterns do not overlap or interfere with one another. As the error space increases the circuit size increases dramatically. Only a few years ago standard alignment tolerances were ± 2 microns; now advanced Intel processes have reduced this dramatically due mostly to the use of advanced projection and step and repeat exposure equipment.

The wafer that is ready for patterning must go through many individual steps before that pattern is complete. First the wafer is baked to remove moisture from its surface and is then treated with chemicals that ensure good resist adhesion. The thick photoresist liquid is then applied and the wafer is spun flat to give a uniform coating, critical for high resolution. The wafer is baked at a low temperature to solidify the resist into gel. It is then exposed with a machine that aligns a mask with the new pattern on it to a previously defined layer. The photo-resist will replicate this pattern on the wafer.

Negative working resists are polymerized by the light and the unexposed resist can be rinsed off with solvents. Positive working resists use photosensitive polymerization inhibitors that allow a chemically reactive developer to remove the exposed areas. The positive resists require much tighter control of exposure and development but yield higher resolution patterns than negative resistance systems.

The wafer is now ready to have its pattern etched. The etch procedure is specialized for each layer to be etched. Wet chemical etchants such as hydrofluoric acid for silicon oxide or phosphoric acid for aluminum are often used for this. The need for

smaller features and tighter control of etched dimensions is increasing the use of plasma etching in fabrication. Here a reactor is run with a partial vacuum into which etchant gases are introduced and an electrical field is applied. This yields a reactive plasma which etches the required layer.

The wafer is now ready for the next process step. Its single journey through the masking process required the careful engineering of mechanics, optics, organic chemistry, inorganic chemistry, plasma chemistry, physics, and electronics.

DIFFUSION

The picture of clean room garbed operators tending furnace tubes glowing cherry red is the one most often associated with IC fabrication. These furnace operations are referred to collectively as diffusion because they employ the principle of solid state diffusion of matter to accomplish their results. In MOS processing, there are three main types of diffusion operations: predepos, drives, and oxidations.

Predeposition, or "predep," is an operation where a dopant is introduced into the furnace from a solid, liquid, or gaseous source and at the furnace temperature (usually 900°C–1200°C) a saturated solution is formed at the silicon surface. The temperature of the furnace, the dopant atom, and rate of introduction are all engineered to give a specific dose of the dopant on the wafer. Once this is completed the wafer is given a drive cycle where the dopant left at the surface by the predep is driven into the wafer by high temperatures. These are generally at different temperatures than the predepos and are designed to give the required junction depth and concentration profile.

Oxidation, the third category, is used at many steps of the process as was shown in the process flow. The temperature and oxidizing ambient can range from 800°C to 1200°C and from pure oxygen to mixtures of oxygen and other gases to steam depending on the type of oxide required. Gate oxides require high dielectric breakdown strength for thin layers (between 0.01 and 0.1 micron) and very tight control over thickness (typically ± 0.005 micron or less than $\pm 1/5,000,000$ inch), while isolation oxides need to be quite thick and because of this their dielectric breakdown strength per unit thickness is much less important.

The properties of the diffused junctions and oxides are key to the performance and reliability of the finished device so the diffusion operations must be extremely well controlled for accuracy, consistency and purity.

ION IMPLANT

Intel's high performance products require such high accuracy and repeatability of dopant control that even the high degree of control provided by diffusion operations is inadequate. However, this limitation has been overcome by replacing critical predepos with ion implantation. In ion implantation, ionized dopant atoms are accelerated by an electric field and implanted directly into the wafer. The acceleration potential determines the depth to which the dopant is implanted.

The charged ions can be counted electrically during implantation giving very tight control over dose. The ion implanters used to perform this are a combination of high vacuum system, ion source, mass spectrometer, linear accelerator, ultra high resolution current integrator, and ion beam scanner. You can see that this important technique requires a host of sophisticated technologies to support it.

THIN FILMS

Thin film depositions make up most of the features on the completed circuit. They include the silicon nitride for defining isolation, polysilicon for the gate and interconnections, the glass for interlayer dielectric, metal for interconnection and external connections, and passivation layers. Thin film depositions are done by two main methods: physical deposition and chemical vapor deposition. Physical deposition is most common for deposition metal. Physical depositions are performed in a vacuum and are accomplished by vaporizing the metal with a high energy electron beam and redepositing it on the wafer or by sputtering it from a target to the wafer under an electric field.

Chemical vapor deposition can be done at atmospheric pressure or under a moderate vacuum. This type of deposition is performed when chemical gases react at the wafer surface and deposit a solid film.

The properties of the diffused junctions and oxides are key to the performance and reliability of the finished device so the diffusion operations must be extremely well controlled for accuracy, consistency and purity.

of the reaction product. These reactors, unlike their general industrial counterparts, must be controlled on a microscale to provide exact chemical and physical properties for thin films such as silicon dioxide, silicon nitride, and polysilicon.

The fabrication of modern memory devices is a long, complex process where each step must be monitored, measured and verified. Developing a totally new manufacturing process for each new product or even product line takes a long time and involves significant risk. Because of this, Intel has developed process families, such as HMOS, on which a wide variety of devices can be made. These families are scalable so that circuits need not be totally redesigned to meet your needs for higher performance.⁽¹⁾ They are evolutionary (HMOS I, HMOS II, HMOS III, CHMOS) so that development time of new processes and products can be reduced without compromising Intel's commitment to consistency, quality, and reliability.

The manufacture of today's MOS memory devices requires a tremendous variety of technologies and manufacturing techniques, many more than could be mentioned here. Each requires a team of experts to design, optimize, control and maintain it. All these people and thousands of others involved in engineering, design, testing and production stand behind Intel's products.

Because of these extensive requirements, most manufacturers have not been able to realize their needs for custom circuits on high performance, high reliability processes. To address this Intel's expertise in this area is now available to industry through the silicon foundry. Intel supplies design rules and support to design and debug circuits. This includes access to Intel's n-well CHMOS technology. Users of the foundry can now benefit from advanced technology without developing processes and IC manufacturing capability themselves.

1 R. Pashley, K. Kokkonen, E. Boleky, R. Jecmen, S. Liu, and W. Owen, "H-MOS Scales Traditional Devices to Higher Performance Level," *Electronics*, August 18, 1977.

Negative working resists are polymerized by the light and the unexposed resist can be rinsed off with solvent. Positive working resists use photoresistive polymers that are insoluble in the developer solution. Positive working resists allow a chemically active developer to remove the exposed areas. The positive resists require much tighter control of exposure and development but yield higher resolution patterns than negative resistance systems.

The water is now ready to have its pattern etched. The etching procedure is specialized for each layer to be etched. Wet chemical etchants such as hydrofluoric acid for silicon oxide or phosphoric acid for aluminum are often used for this. The need for

Memory Datasheets

6

Memory Datasets

6



27C64/87C64 64K (8K x 8) CHMOS PRODUCTION AND UV ERASABLE PROMS

Automotive

- **Extended Automotive Temperature Range**
— -40°C to $+125^{\circ}\text{C}$
- **CHMOS Microcontroller and Microprocessor Compatible**
 - 87C64-Integrated Address Latch
 - Universal 28 Pin Memory Site, 2-line Control
- **Low Power Consumption**
 - 100 μA Maximum Standby Current
- **High Performance Speeds**
 - 200 ns Maximum Access Time
- **Noise Immunity Features**
 - $\pm 10\%$ V_{CC} Tolerance
 - Maximum Latch-up Immunity Through EPI Processing
- **New Quick-Pulse Programming™ Algorithm** (1 second programming)
 - Use with Plastic DIP and PLCC EPROMs
- **Available in 28-Pin Cerdip and Plastic DIP Pkg. and 32-Lead PLCC Pkg.**
(See Packaging Spec, Order #231369)

Intel's 27C64 and 87C64 CHMOS EPROMs are 64K bit 5V only memories organized as 8192 words of 8 bits. They employ advanced CHMOS*II-E circuitry for systems requiring low power, high performance speeds, and immunity to noise. The 87C64 has been optimized for multiplexed bus microcontroller and microprocessor compatibility while the 27C64 has a non-multiplexed addressing interface and is plug compatible with the standard Intel 2764A (HMOS II-E).

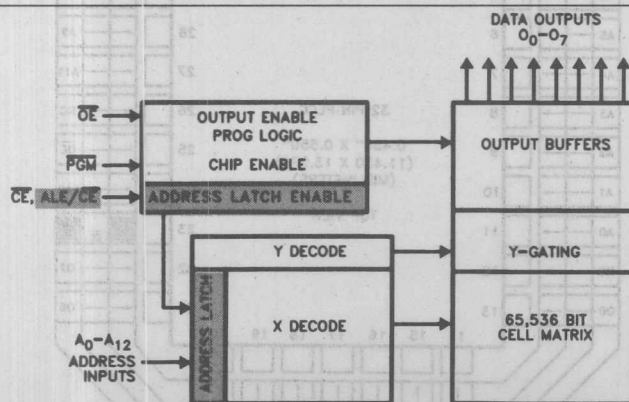
The 27C64 and 87C64 are offered in both a ceramic DIP, Plastic DIP, and Plastic Leaded Chip Carrier (PLCC) Packages. Cerdip packages provide flexibility in prototyping and R&D environments, whereas Plastic DIP and PLCC EPROMs provide optimum cost effectiveness in production environments. A new Quick-Pulse Programming™ Algorithm is employed on Plastic DIP and PLCC devices which may speed up programming by as much as one hundred times. In the absence of Quick-Pulse compatible programming equipment and with Cerdip devices, the intelligent Programming Algorithm may be utilized.

The 87C64 incorporates an address latch on the address pins to minimize chip count in multiplexed bus systems. Designers can eliminate the address latch by tying address and data pins of the 87C64 directly to the processor's multiplexed address/data pins. On the falling edge of the ALE input (ALE/CE), address information at the address inputs (A_0 – A_{12}) of the 87C64 is latched internally. The address inputs are then ignored as data information is passed on the same bus.

The highest degree of protection against latch-up is achieved through Intel's unique EPI processing. Prevention of latch-up is provided for stresses up to 100 mA on address and data pins from -1V to $V_{\text{CC}} + 1\text{V}$.

In order to meet the rigorous environmental requirements of automotive applications, Intel offers the 27C64/87C64 in extended Automotive temperature range. Operational characteristics are guaranteed over the range of -40°C to $+125^{\circ}\text{C}$ ambient.

*HMOS and CHMOS are patented processes of Intel Corporation.



Shaded Areas represent the 87C64 version

290119-1

Figure 1. Block Diagram

Pin Names

A ₀ -A ₁₂	ADDRESSES
O ₀ -O ₇	OUTPUTS
OE	OUTPUT ENABLE
CE	CHIP ENABLE
ALE/CE	ADDRESS LATCH ENABLE /CHIP ENABLE
PGM	PROGRAM STROBE
N.C.	NO CONNECT
D.U.	DON'T USE

27C64/87C64

P27C64

27256	27128	2732A	2716	2716	2732A	27128	27256
V _{pp}	V _{pp}	V _{pp}	V _{pp}	V _{pp}	V _{pp}	V _{pp}	V _{pp}
A ₁₂	A ₁₂	A ₁₂	A ₁₂	A ₁₂	A ₁₂	A ₁₂	A ₁₂
A ₇	A ₇	A ₇	A ₇	A ₇	A ₇	A ₇	A ₇
A ₆	A ₆	A ₆	A ₆	A ₆	A ₆	A ₆	A ₆
A ₅	A ₅	A ₅	A ₅	A ₅	A ₅	A ₅	A ₅
A ₄	A ₄	A ₄	A ₄	A ₄	A ₄	A ₄	A ₄
A ₃	A ₃	A ₃	A ₃	A ₃	A ₃	A ₃	A ₃
A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂
A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁
A ₀	A ₀	A ₀	A ₀	A ₀	A ₀	A ₀	A ₀
O ₀	O ₀	O ₀	O ₀	O ₀	O ₀	O ₀	O ₀
O ₁	O ₁	O ₁	O ₁	O ₁	O ₁	O ₁	O ₁
O ₂	O ₂	O ₂	O ₂	O ₂	O ₂	O ₂	O ₂
Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd

NOTE:

Intel "Universal Site" Compatible EPROM Pin Configurations are shown in the adjacent blocks to 27C64 Pins.

Shaded Areas represent the 87C64 version

Figure 2. Pin Configuration

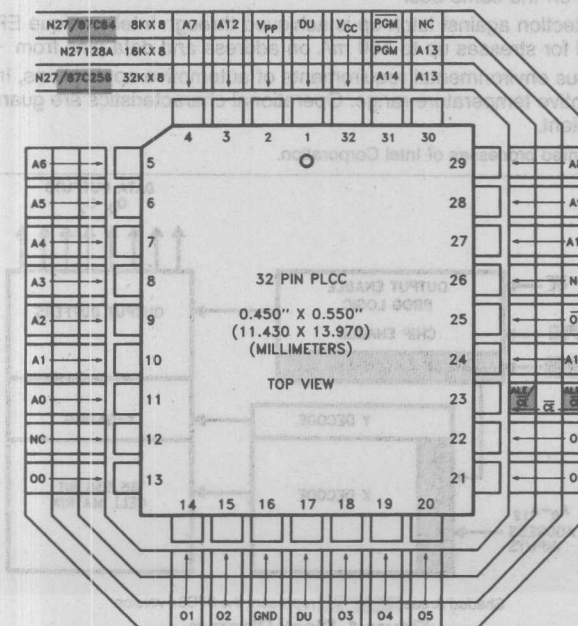


Figure 3. PLCC(N) Lead Configuration

Extended Temperature (Express) EPROMs

The Intel EXPRESS EPROM family is a series of electrically programmable read only memories which have received additional processing to enhance product characteristics. EXPRESS processing is available for several densities of EPROM, allowing the choice of appropriate memory size to match system applications.

EXPRESS EPROM products are available with 168 ± 8 hour, 125°C dynamic burn-in using Intel's standard bias configuration. This process exceeds or meets most industry specifications of burn-in. The standard EXPRESS EPROM operating temperature range is 0°C to 70°C. Extended operating temperature range (-40°C to +85°C) EXPRESS products are available along with automotive temperature range (-40°C to +125°C) products.

AUTOMOTIVE AND EXPRESS EPROM PRODUCT FAMILY

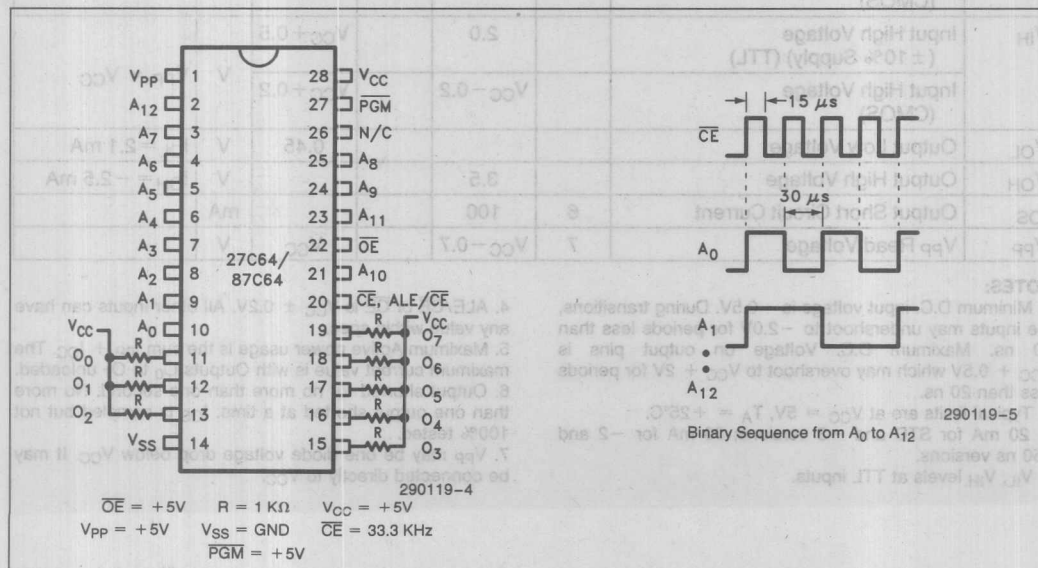
PRODUCT DEFINITIONS

Type	Operating Temperature (°C)	Burn-in 125°C (hr)
Q	0 to +70	168 ± 8
T	-40 to +85	NONE
L	-40 to +85	168 ± 8
A	-40 to +125	NONE
B	-40 to +125	168 ± 8

AUTOMOTIVE AND EXPRESS OPTIONS

27C64/87C64 Versions

Packaging Options			
Speed Versions	Cerdip	PLCC	Plastic DIP
-1	T, L, Q	T	T
-15	T, L, Q	T	T
-2	T, L, Q, A, B	T, A	T, A
-20	T, L, Q, A, B	T	T
-STD	T, L, Q, A, B	T, A	T, A
-25	T, L, Q, A, B	T	T
-3	T, L, Q, A, B	T, A	T, A
-30	T, L, Q, A, B	T	T



Burn-In Bias and Timing Diagrams

Operating Temperature	
During Read	−40°C to +125°C
Temperature Under Bias	−40°C to +125°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to Ground	−2.0V to 7V(1)
Voltage on Pin A ₉ with Respect to Ground	−2.0V to +13.5V(1)
V _{PP} Supply Voltage with Respect to Ground	
During Programming	−2.0V to +14V(1)
V _{CC} Supply Voltage with Respect to Ground	−2.0V to +7.0V(1)

Maximum Thermal Resistance

Junction to Ambient (θ_{JA}):	Cerdip	46°C/W
	Plastic	69°C/W
	PLCC	64°C/W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

READ OPERATION D.C. CHARACTERISTICS −40°C ≤ T_A ≤ +125°C

Symbol	Parameter	Notes	Min	Typ(2)	Max	Unit	Test Condition
I _{LI}	Input Leakage Current			0.01	±1.0	μA	V _{IN} = 0V, 5.5V
I _{LO}	Output Leakage Current			0.01	±10	μA	V _{OUT} = 0V, 5.5V
I _{PP1}	V _{PP} Current Read	5			100	μA	V _{PP} = V _{CC}
I _{SB}	V _{CC} Current Standby with Inputs—	CMOS	4		100	μA	$\overline{CE} = V_{IH}$
		TTL	3		1.0	mA	
I _{CC1}	V _{CC} Current Active (mA)	TTL			20, 30	mA	$\overline{OE} = \overline{CE} = V_{IL}$
	V _{CC} Current Active at High Temperature	TTL			20, 30	mA	$\overline{OE} = \overline{CE} = V_{IL}$ V _{PP} = V _{CC} , T _{ambient} = 125°C
V _{IL}	Input Low Voltage (±10% Supply) (TTL)		−0.5		0.8	V	V _{PP} = V _{CC}
	Input Low Voltage (CMOS)		−0.2		0.2		
V _{IH}	Input High Voltage (±10% Supply) (TTL)		2.0		V _{CC} + 0.5	V	V _{PP} = V _{CC}
	Input High Voltage (CMOS)		V _{CC} − 0.2		V _{CC} + 0.2		
V _{OL}	Output Low Voltage				0.45	V	I _{OL} = 2.1 mA
V _{OH}	Output High Voltage		3.5			V	I _{OH} = −2.5 mA
I _{OS}	Output Short Circuit Current	6	100			mA	
V _{PP}	V _{PP} Read Voltage	7	V _{CC} − 0.7		V _{CC}	V	

NOTES:

1. Minimum D.C. input voltage is −0.5V. During transitions, the inputs may undershoot to −2.0V for periods less than 20 ns. Maximum D.C. Voltage on output pins is V_{CC} + 0.5V which may overshoot to V_{CC} + 2V for periods less than 20 ns.
2. Typical limits are at V_{CC} = 5V, T_A = +25°C.
3. 20 mA for STD and −3 versions; 30 mA for −2 and 150 ns versions.
V_{IL}, V_{IH} levels at TTL inputs.

4. ALE/ \overline{CE} or \overline{CE} is V_{CC} ± 0.2V. All other inputs can have any value within spec.
5. Maximum Active power usage is the sum I_{PP} + I_{CC}. The maximum current value is with Outputs O₀ to O₇ unloaded.
6. Output shorted for no more than one second. No more than one output shorted at a time. I_{OS} is sampled but not 100% tested.
7. V_{PP} may be one diode voltage drop below V_{CC}. It may be connected directly to V_{CC}.

READ OPERATION

A.C. CHARACTERISTICS 27C64(1) $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$

Versions (3)		$V_{CC} \pm 5\%$	27C64-2		27C64		27C64-3		Unit
			N27C64-2	N27C64	N27C64-3				
			P27C64-2	P27C64	P27C64-3				
		$V_{CC} \pm 10\%$	27C64-20		27C64-25		27C64-30		
Symbol	Characteristic	Min	Max	Min	Max	Min	Max		
t_{ACC}	Address to Output Delay		200		250		300	ns	
t_{CE}	\overline{CE} to Output Delay		200		250		300	ns	
t_{OE}	\overline{OE} to Output Delay		75		100		120	ns	
$t_{DF}^{(2)}$	\overline{OE} High to Output High Z		55		60		105	ns	
$t_{OH}^{(2)}$	Output Hold from Addresses, \overline{CE} or \overline{OE} Change-Whichever is First	0		0		0		ns	

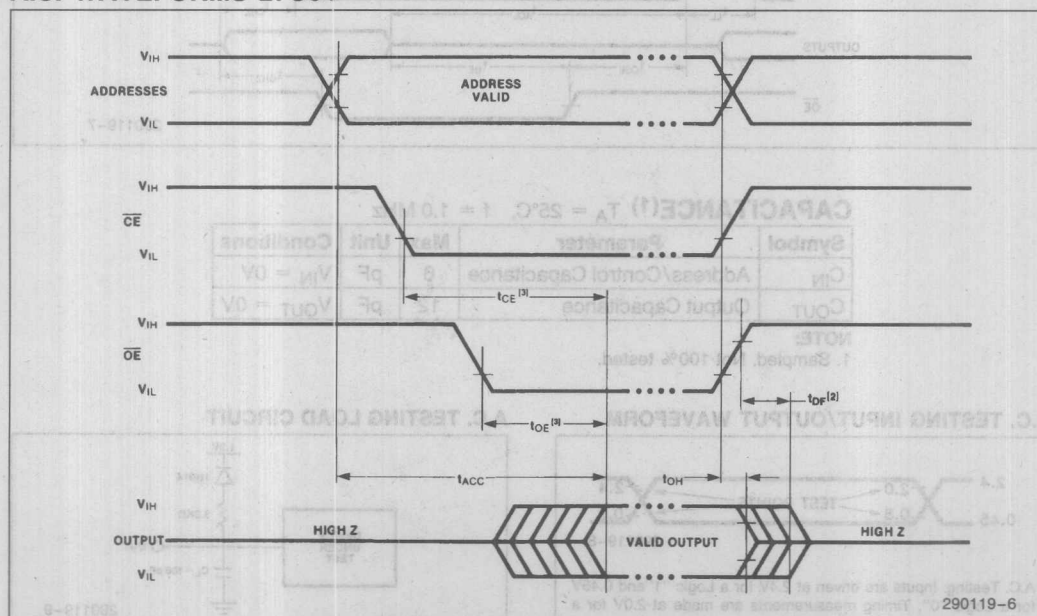
NOTES:

1. A.C. characteristics tested at $V_{IH} = 2.4\text{V}$ and $V_{IL} = 0.45\text{V}$.
Timing measurements made at $V_{OL} = 0.8\text{V}$ and $V_{OH} = 2.0\text{V}$.
2. Guaranteed and sampled.
3. Part Number Prefixes: No prefix = Cerdip; P = Plastic DIP; N = PLCC.

A.C. CONDITIONS OF TEST

Input Rise and Fall Times (10% to 90%) 10 ns
 Input Pulse Levels 0.45V to 2.4V
 Input Timing Reference Level 0.8V and 2.0V
 Output Timing Reference Level 0.8V and 2.4V

A.C. WAVEFORMS 27C64



NOTES:

1. Typical values are for $T_A = 25^{\circ}\text{C}$ and nominal supply voltages.
2. This parameter is only sampled and is not 100% tested.
3. \overline{OE} may be delayed up to $t_{CE} - t_{OE}$ after the falling edge of \overline{CE} without impact on t_{CE} .

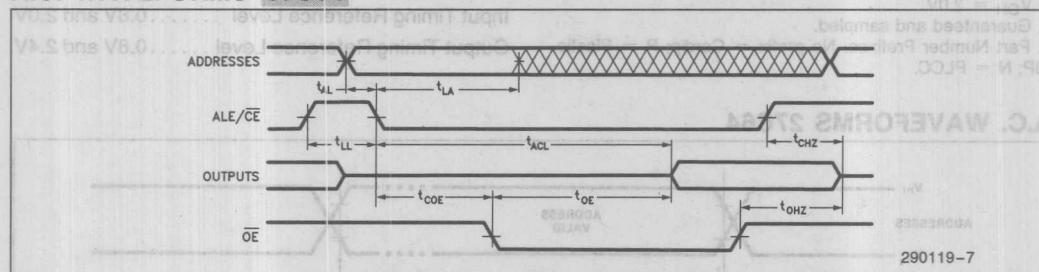
A.C. CHARACTERISTICS 87C64(1) $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$

Versions (3)		V _{CC} ± 5%		87C64-2 N87C64-2		87C64 N87C64		87C64-3 N87C64-3		Unit
		V _{CC} ± 10%		87C64-20		87C64-25		87C64-30		
Symbol	Parameter	Min	Max	Min	Max	Min	Max	Min	Max	
t _{LL}	Chip Deselect Width	50		60				75		ns
t _{AL}	Address to $\overline{\text{CE}}$ -Latch Set-up	20		25				30		ns
t _{LA}	Address Hold from $\overline{\text{CE}}$ -LATCH	45		50				60		ns
t _{ACL}	CE-Latch Access Time		200		250				300	ns
t _{OE}	Output Enable to Output Valid		75		100				120	ns
t _{COE}	ALE/ $\overline{\text{CE}}$ to Output Enable	45		50				60		ns
t _{CHZ} ⁽²⁾	Chip Deselect to Output in High Z		50		60				75	ns
t _{OHZ} ⁽²⁾	Output Disable to Output in High Z		50		60				75	ns

NOTES:

1. A.C. characteristics tested at $V_{IH} = 2.4\text{V}$ and $V_{IL} = 0.45\text{V}$.
Timing measurements made at $V_{OL} = 0.8\text{V}$ and $V_{OH} = 2.0\text{V}$.
2. Guaranteed and sampled.
3. Model Number Prefixes: No prefix = Cerdip; P = Plastic DIP; N = PLCC.

A.C. WAVEFORMS 87C64



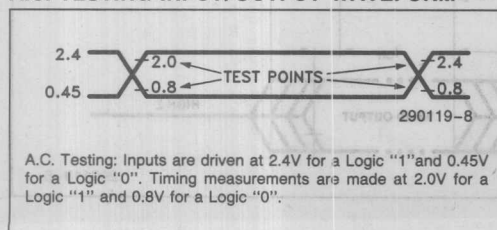
CAPACITANCE(1) $T_A = 25^{\circ}\text{C}$, $f = 1.0\text{ MHz}$

Symbol	Parameter	Max	Unit	Conditions
C_{IN}	Address/Control Capacitance	6	pF	$V_{IN} = 0\text{V}$
C_{OUT}	Output Capacitance	12	pF	$V_{OUT} = 0\text{V}$

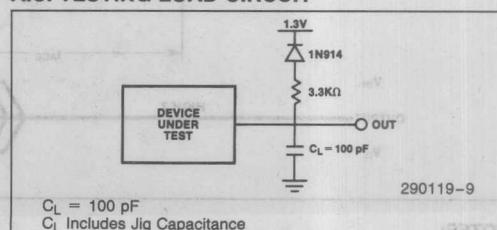
NOTE:

1. Sampled. Not 100% tested.

A.C. TESTING INPUT/OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT



DEVICE OPERATION

The modes of operation of the 27C64/87C64 are listed in Table 1. A single 5V power supply is required in the read mode. All inputs are TTL levels except for V_{PP} and 12V on A_9 for intelligent Identifier mode.

Table 1. Mode Selection for 27C64 and 87C64

Mode	Pins	ALE/CE CE	OE	PGM (7)	A_9	A_0	V_{PP}	V_{CC}	Outputs
Read		V_{IL}	V_{IL}	V_{IH}	X ⁽¹⁾	X	V_{CC}	5.0V	D_{OUT}
Output Disable		V_{IL}	V_{IH}	V_{IH}	X	X	V_{CC}	5.0V	High Z
Standby		V_{IH}	X	X	X	X	V_{CC}	5.0V	High Z
Programming		V_{IL}	V_{IH}	V_{IL}	X	X	(4)	(4)	D_{IN}
Program Verify		V_{IL}	V_{IL}	V_{IH}	X	X	(4)	(4)	D_{OUT}
Program Inhibit		V_{IH}	X	X	X	X	(4)	(4)	HIGH Z
intelligent Identifier ⁽³⁾ -Manufacturer		V_{IL}	V_{IL}	V_{IH}	V_H ⁽²⁾	V_{IL}	V_{CC}	V_{CC}	89 H (6) 88 H (6)
intelligent Identifier ⁽³⁾ -27C64		V_{IL}	V_{IL}	V_{IH}	V_H ⁽²⁾	V_{IH}	V_{CC}	V_{CC}	07 H
intelligent Identifier ^(3, 5) -87C64		V_{IL}	V_{IL}	V_{IH}	V_H ⁽²⁾	V_{IH}	V_{CC}	V_{CC}	37 H

NOTES:

1. X can be V_{IL} or V_{IH} .
2. $V_H = 12.0V \pm 0.5V$.
3. $A_1-A_8, A_{10}-12 = V_{IL}$.
4. See Table 2 for V_{CC} and V_{PP} voltages.
5. ALE/CE has to be toggled in order to latch in the addresses and read the signature codes.
6. The Manufacturer's identifier reads 89H for CerDip devices; 88H for Plastic DIP and PLCC devices.
7. In Read Mode tie PGM to V_{CC} .

Read Mode: 27C64

The 27C64 has two control functions, both of which must be logically active in order to obtain data at the outputs. Chip Enable (\overline{CE}) is the power control and should be used for device selection. Output enable (\overline{OE}) is the output control and should be used to gate data from the output pins. Assuming that addresses are stable, the address access time (t_{ACC}) is equal to the delay from \overline{CE} to output (t_{OE}). Data is available at the outputs after a delay of t_{OE} from the falling edge of \overline{OE} , assuming that \overline{CE} has been low and addresses have been stable for at least $t_{ACC}-t_{OE}$.

The 87C64 internal address latch is directly enabled through the use of the ALE/CE line. As the transition occurs on the ALE/CE from the TTL high to the low state, the last address presented at the address pins is retained. Data is then enabled onto the bus from the EPROM via the \overline{OE} pin.

Read Mode: 87C64

The 87C64 was designed to reduce the hardware interface requirements when incorporated in processor systems with multiplexed address-data busses. Chip count (and therefore power and board space) can be minimized when the 87C64 is designed as shown in Figure 4. The processor's multiplexed bus (AD_{0-7}) is tied to both address and data pins of the 87C64. All address inputs of the 87C64 are latched when ALE/CE is brought low, thus eliminating the need for a separate address latch.

The 87C64 internal address latch is directly enabled through the use of the ALE/CE line. As the transition occurs on the ALE/CE from the TTL high to the low state, the last address presented at the address pins is retained. Data is then enabled onto the bus from the EPROM via the OE pin.

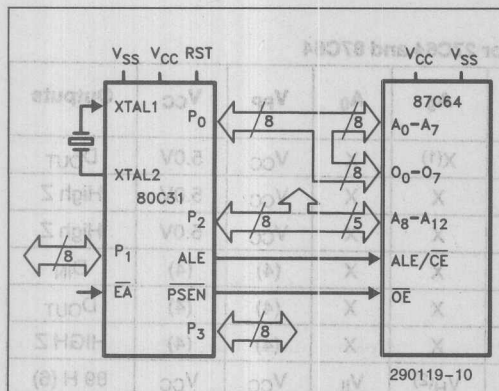


Figure 4. 80C31 with 87C64
System Configuration

Standby Mode

The 27C64 and 87C64 have Standby modes which reduce the maximum V_{CC} current to 100 μA . Both are placed in the Standby mode when CE or ALE/CE are in the CMOS-high state. When in the Standby mode, the outputs are in a high impedance state, independent of the OE input.

Two Line Output Control

Because EPROMs are usually used in larger memory arrays, Intel has provided 2 control lines which accommodate this multiple memory connection. The two control lines allow for:

- the lowest possible memory power dissipation, and
- complete assurance that output bus contention will not occur.

To use these two control lines most efficiently, CE (or ALE/CE) should be decoded and used as the primary device selecting function, while OE should be made a common connection to all devices in the array and connected to the READ line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are active only when data is desired from a particular memory device.

SYSTEM CONSIDERATIONS

The power switching characteristics of EPROMs require careful decoupling of the devices. The supply current, I_{CC} , has three segments that are of interest to the system designer—the standby current level, the active current level, and the transient current peaks that are produced by the falling and rising edges of Chip Enable. The magnitude of these transient and inductive current peaks is dependent on the output capacitive and inductive loading of the device. The associated transient voltage peaks can be suppressed by complying with Intel's Two-Line Control, and by properly selected decoupling capacitors. It is recommended that a 0.1 μF ceramic capacitor be used on every device between V_{CC} and GND. This should be a high frequency capacitor for low inherent inductance and should be placed as close to the device as possible. In addition, a 4.7 μF bulk electrolytic capacitor should be used between V_{CC} and GND for every eight devices. The bulk capacitor should be located near where the power supply is connected to the array. The purpose of the bulk capacitor is to overcome the voltage droop caused by the inductive effect of PC board-traces.

PROGRAMMING MODES

Caution: Exceeding 14V on V_{PP} will permanently damage the device.

Initially, and after each erasure, all bits of the EPROM are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The device is in the programming mode when V_{PP} is raised to its programming voltage (See Table 2) and CE (or ALE/CE) and PGM are both at TTL low and OE = V_{IH} . The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

Program Inhibit

Programming of multiple EPROMs in parallel with different data is easily accomplished by using the Program Inhibit mode. A high-level CE (or ALE/CE) or PGM input inhibits the other devices from being programmed.

Except for \overline{OE} (or ALE/\overline{CE}), all like inputs (including \overline{OE}) of the parallel EPROMs may be common. A TTL low-level pulse applied to the \overline{PGM} input with V_{PP} at its programming voltage and \overline{CE} (or ALE/\overline{CE}) = V_{IL} will program the selected device.

Program Verify

A verify (read) should be performed on the programmed bits to determine that they have been correctly programmed. The verify is performed with \overline{OE} and \overline{CE} (or ALE/\overline{CE}) at V_{IL} , \overline{PGM} at V_{IH} , and V_{CC} and V_{PP} at their programming voltages. Data should be verified a minimum of t_{OE} after the falling edge of \overline{OE} .

intelligent Identifier™ Mode

The intelligent Identifier Mode allows the reading out of a binary code from an EPROM that will identify its manufacturer and type. This mode is intended for use by programming equipment for the purpose of automatically matching the device to be programmed with its corresponding programming algorithm. This mode is functional in the $25^{\circ}\text{C} \pm 5^{\circ}\text{C}$ ambient temperature range that is required when programming the device.

To activate this mode, the programming equipment must force 11.5V to 12.5V on address line A9 of the EPROM. Two identifier bytes may then be sequenced from the device outputs by toggling address line A0 from V_{IL} to V_{IH} . All other address lines must be held at V_{IL} during the intelligent Identifier Mode.

Byte 0 ($A0 = V_{IL}$) represents the manufacturer code and byte 1 ($A0 = V_{IH}$) the device identifier code. These two identifier bytes are given in Table 1. ALE/\overline{CE} of the 87C64 has to be toggled in order to latch in the addresses and read the Signature Codes.

INTEL EPROM PROGRAMMING SUPPORT TOOLS

Intel offers a full line of EPROM Programmers providing state-of-the-art programming for Intel programmable devices. The modular architecture of Intel's EPROM programmers allows you to add new support as it becomes available, with very low cost add-ons. For example, even the earliest users of the iUP-FAST 27/K module may take advantage of Intel's new Quick-Pulse Programming Algorithm, the fastest in the industry.

Intel EPROM programmers may be controlled from a host computer using Intel's PROM Programming

software (iPPS). iPPS makes programming easy for a growing list of industry standard hosts, including the IBM PC, XT, AT, and PC DOS compatibles, Inteltec Development Systems, Intel's iPDS Personal Development System, and the Intel Network Development System (iNDS-II). Stand-alone operation is also available, including device previewing, editing, programming, and download of programming data from any source over an RS232C port.

For further details consult the EPROM Programming section of the Development Systems Handbook.

ERASURE CHARACTERISTICS (FOR Cerdip EPROMs)

The erasure characteristics are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms (\AA). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000 \AA range. Data shows that constant exposure to room level fluorescent lighting could erase the EPROM in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the device is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the window to prevent unintentional erasure.

The recommended erasure procedure is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (\AA). The integrated dose (i.e., UV intensity \times exposure time) for erasure should be a minimum of 15 Wsec/cm². The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 $\mu\text{W}/\text{cm}^2$ power rating. The EPROM should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated dose an EPROM can be exposed to without damage is 7258 Wsec/cm² (1 week @ 12000 $\mu\text{W}/\text{cm}^2$). Exposure of the device to high intensity UV light for longer periods may cause permanent damage.

CHMOS NOISE CHARACTERISTICS

Special EPI processing techniques have enabled Intel to build CHMOS with features adding to system reliability. These include input/output protection to latch-up. Each of the data and address pins will not latch-up with currents up to 100 mA and voltages from -1V to $V_{CC} + 1\text{V}$.

Additionally, the V_{PP} (programming) pin is designed to resist latch-up to the 14V maximum device limit.

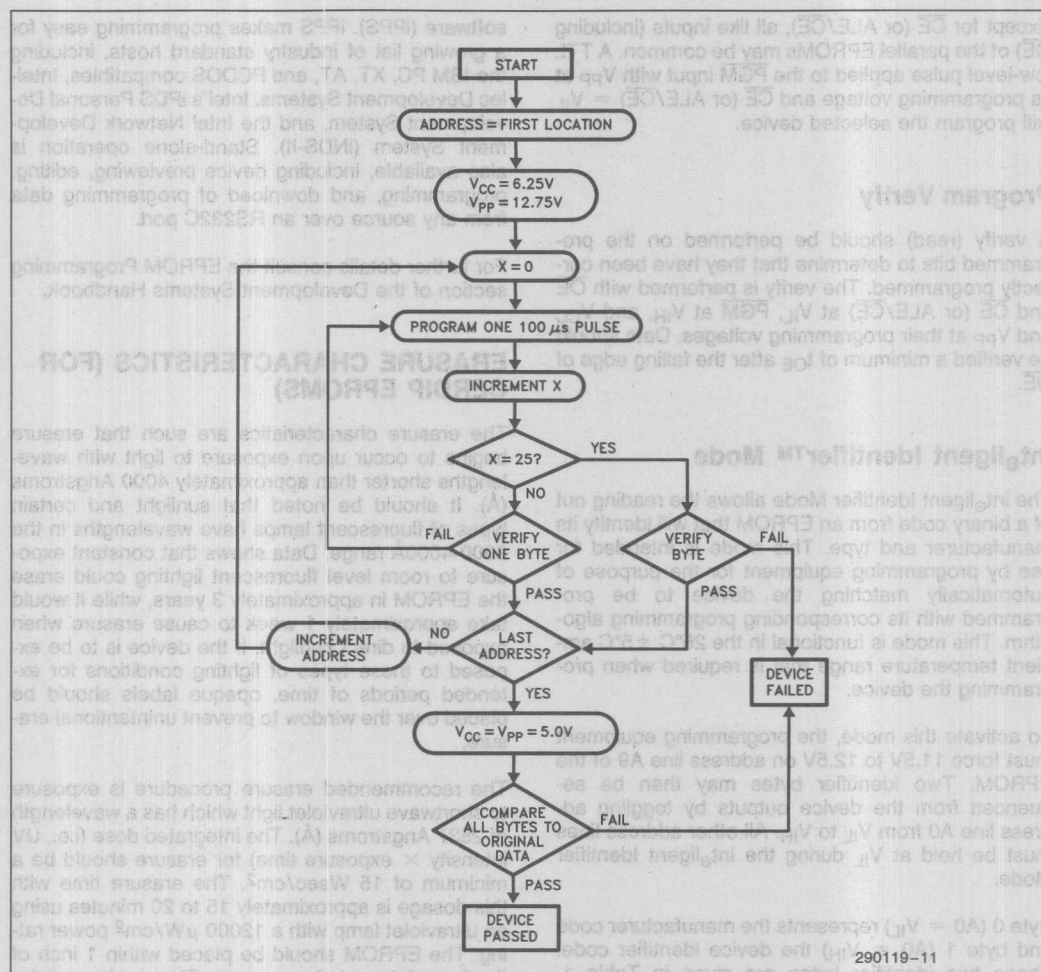


Figure 5. Quick-Pulse Programming™ Algorithm

Quick-Pulse Programming™ Algorithm (for Plastic DIP and PLCC EPROMs)

Intel's Plastic DIP and PLCC EPROMs can now be programmed using the Quick-Pulse Programming Algorithm, developed by Intel to substantially reduce the throughput time in the production environment. This algorithm allows these devices to be programmed in under one second, almost a hundred fold improvement over previous algorithms. Actual programming time is a function of the PROM programmer being used.

The Quick-Pulse Programming Algorithm uses initial pulses of 100 microseconds followed by a byte verification to determine when the address byte has been successfully programmed. Up to 25 100 μs

pulses per byte are provided before a failure is recognized. A flowchart of the Quick-Pulse Programming Algorithm is shown in Figure 5.

For the Quick Pulse Programming Algorithm, the entire sequence of programming pulses and byte verifications is performed at $V_{CC} = 6.25V$ and $V_{PP} = 12.75V$. When programming of the EPROM has been completed, all bytes should be compared to the original data with $V_{CC} = V_{PP} = 5.0V$.

In addition to the Quick-Pulse Programming Algorithm, PLCC EPROMs are also compatible with Intel's intelligent Programming Algorithm.

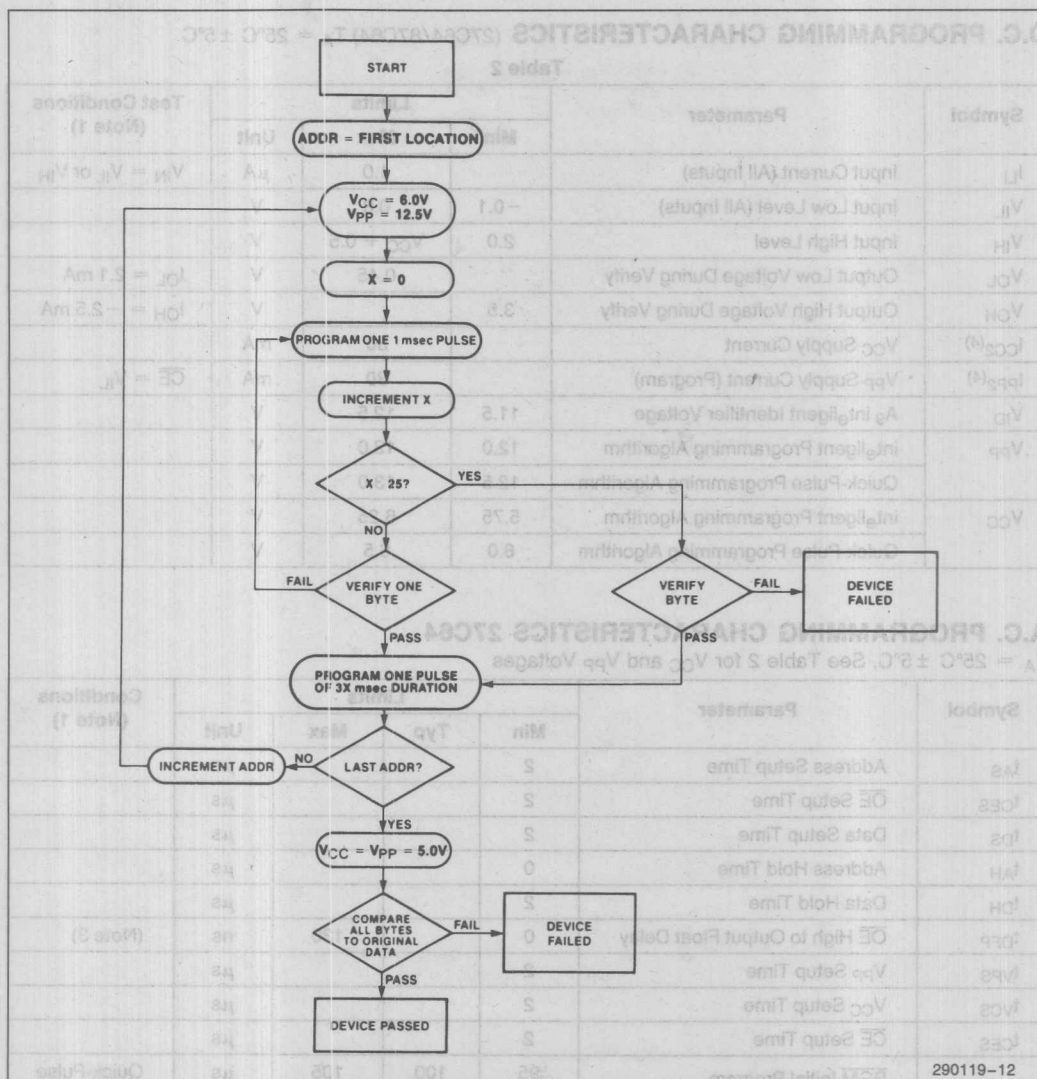


Figure 6. intelligent Programming™ Flowchart

intelligent Programming™ Algorithm

The intelligent Programming Algorithm has been a standard in the industry for the past few years. A flowchart of the intelligent Programming Algorithm is shown in Figure 6.

The intelligent Programming Algorithm utilizes two different pulse types: initial and overprogram. The duration of the initial pulse(s) is one millisecond, which will then be followed by a larger overprogram pulse of length 3X msec. X is an iteration counter

and is equal to the number of the initial one millisecond pulses applied to a particular location, before a correct verify occurs. Up to 25 one-millisecond pulses per byte are provided for before the overprogram pulse is applied.

The entire sequence of program pulses and byte verifications is performed at $V_{CC} = 6.0V$ and $V_{PP} = 12.5V$. When the intelligent Programming cycle has been completed, all bytes should be compared to the original data with $V_{CC} = V_{PP} = 5.0V$.

D.C. PROGRAMMING CHARACTERISTICS (27C64/87C64) $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$

Table 2

Symbol	Parameter	Limits			Test Conditions (Note 1)
		Min	Max	Unit	
I_{LI}	Input Current (All Inputs)		1.0	μA	$V_{IN} = V_{IL} \text{ or } V_{IH}$
V_{IL}	Input Low Level (All Inputs)	-0.1	0.8	V	
V_{IH}	Input High Level	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage During Verify		0.45	V	$I_{OL} = 2.1 \text{ mA}$
V_{OH}	Output High Voltage During Verify	3.5		V	$I_{OH} = -2.5 \text{ mA}$
$I_{CC2}^{(4)}$	V_{CC} Supply Current		30	mA	
$I_{PP2}^{(4)}$	V_{PP} Supply Current (Program)		30	mA	$\overline{CE} = V_{IL}$
V_{ID}	A_9 intelligent Identifier Voltage	11.5	12.5	V	
V_{PP}	intelligent Programming Algorithm	12.0	13.0	V	
	Quick-Pulse Programming Algorithm	12.5	13.0	V	
V_{CC}	intelligent Programming Algorithm	5.75	6.25	V	
	Quick-Pulse Programming Algorithm	6.0	6.5	V	

A.C. PROGRAMMING CHARACTERISTICS 27C64

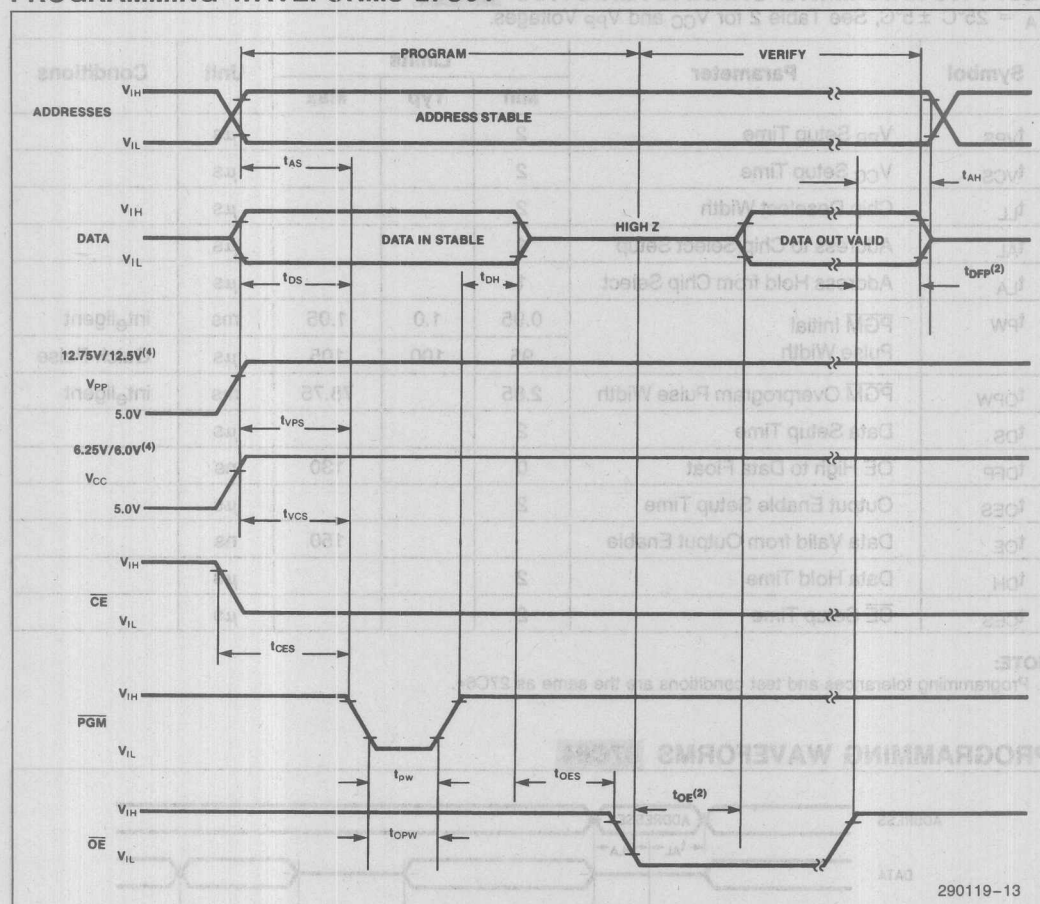
$T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$, See Table 2 for V_{CC} and V_{PP} Voltages

Symbol	Parameter	Limits				Conditions (Note 1)
		Min	Typ	Max	Unit	
t_{AS}	Address Setup Time	2			μs	
t_{OES}	\overline{OE} Setup Time	2			μs	
t_{DS}	Data Setup Time	2			μs	
t_{AH}	Address Hold Time	0			μs	
t_{DH}	Data Hold Time	2			μs	
t_{DFP}	\overline{OE} High to Output Float Delay	0		130	ns	(Note 3)
t_{VPS}	V_{PP} Setup Time	2			μs	
t_{VCS}	V_{CC} Setup Time	2			μs	
t_{CES}	\overline{CE} Setup Time	2			μs	
t_{PW}	PGM Initial Program Pulse Width	95	100	105	μs	Quick-Pulse
	Pulse Width	0.95	1.0	1.05	ms	intelligent
t_{OPW}	PGM Overprogram Pulse Width	2.85		78.75	ms	(Note 2)
t_{OE}	Data Valid from \overline{OE}			150	ns	

NOTES:

- V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP} .
- The length of the overprogram pulse (intelligent Programming Algorithm) may vary from 2.85 msec to 78.75 msec as a function of the iteration counter value X.
- This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.
- The maximum current value is with outputs O_0 to O_7 Unloaded.

PROGRAMMING WAVEFORMS 27C64



NOTES:

1. The Input Timing Reference Level is 0.8V for V_{IL} and 2V for a V_{IH} .
2. t_{OE} and t_{DFF} are characteristics of the device but must be accommodated by the programmer.
3. When programming the 27C64, a 0.1 μF capacitor is required across V_{PP} and ground to suppress spurious voltage transients which can damage the device.
4. 12.75V V_{PP} /6.25V V_{CC} for Quick-Pulse Programming Algorithm; 12.5V V_{PP} /6.0V V_{CC} for Intelligent Programming Algorithm.

A.C. PROGRAMMING CHARACTERISTICS 87C64

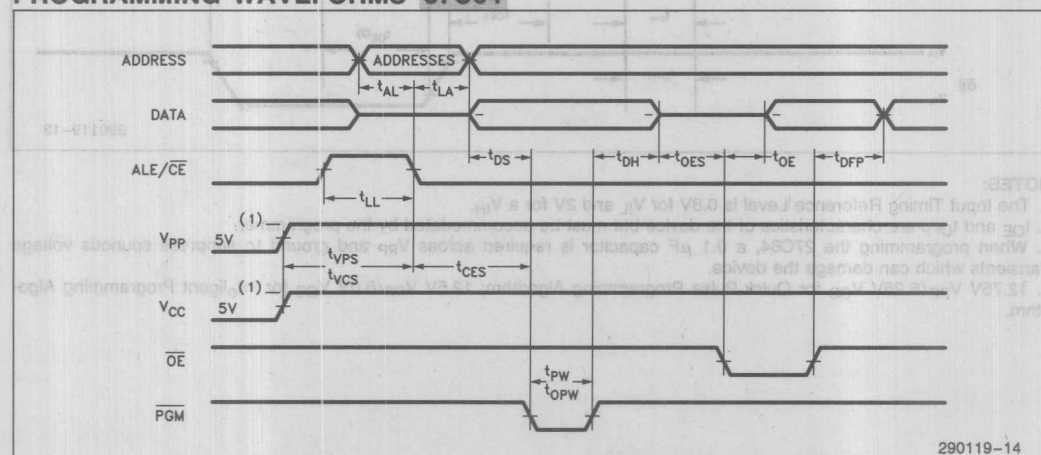
T_A = 25°C ±5°C, See Table 2 for V_{CC} and V_{PP} Voltages.

Symbol	Parameter	Limits			Unit	Conditions
		Min	Typ	Max		
t _{VPS}	V _{PP} Setup Time	2			μs	
t _{VCS}	V _{CC} Setup Time	2			μs	
t _{LL}	Chip Deselect Width	2			μs	
t _{AL}	Address to Chip Select Setup	1			μs	
t _{LA}	Address Hold from Chip Select	1			μs	
t _{PW}	PGM Initial Pulse Width	0.95	1.0	1.05	ms	intelligent
		95	100	105	μs	Quick-Pulse
t _{OPW}	PGM Overprogram Pulse Width	2.85		78.75	ms	intelligent
t _{DS}	Data Setup Time	2			μs	
t _{DFP}	OE High to Data Float	0		130	ns	
t _{OES}	Output Enable Setup Time	2			μs	
t _{OE}	Data Valid from Output Enable			150	ns	
t _{DH}	Data Hold Time	2			μs	
t _{CES}	CE Setup Time	2			μs	

NOTE:

1. Programming tolerances and test conditions are the same as 27C64.

PROGRAMMING WAVEFORMS 87C64



NOTE:

1. 12.75V V_{PP} & 6.25V V_{CC} for Quick-Pulse Programming Algorithm; 12.5V V_{PP} & 6.0V V_{CC} for intelligent Programming Algorithm.

27C128 128K (16K x 8) CHMOS PRODUCTION AND UV ERASABLE PROMs

Automotive

- **Extended Automotive Temperature Range** — -40°C to $+125^{\circ}\text{C}$
- **CHMOS Microcontroller and Microprocessor Compatible**
- **Low Power Consumption** — $100\ \mu\text{A}$ Maximum Standby Current
- **Maximum Latch-Up Immunity Through EPI Processing**
 - $\pm 1\text{V}$ Input Protection
 - 14V V_{PP} Protection
- **High Performance** — $200\ \text{ns}$ Access Time
- **Quick-Pulse Programming™ Algorithm Allows Rapid, Automated Programming** — 2 Second Throughput
- **Available in 28-Pin Cerdip and Plastic DIP and 32-Lead PLCC Packages**
(See Packaging Spec. Order #231369)

Intel's 27C128 CHMOS EPROM is a 128K bit 5V-only memory, organized as 16,384 words of 8 bits each. The 27C128 is ideal for systems requiring low power, high performance, and noise immunity due to its CHMOS™II-E processing, and it is pin compatible with the standard Intel 27128A and 27128B.

The 27C128 is offered in Ceramic DIP, Plastic DIP, and Plastic Leaded Chip Carrier (PLCC) Packages. Cerdip packages provide flexibility in prototyping and R & D environments while the Plastic DIP and PLCC packages are most cost effective in production environments. The Quick-Pulse Programming™ Algorithm improves programming speed by as much as one hundred times over older algorithms, further reducing costs for system manufacturers.

Intel's unique EPI processing provides excellent latch-up immunity. Prevention of latch-up is guaranteed for stresses up to 100 mA on address and data pins from -1V to $V_{\text{CC}} + 1\text{V}$ and for V_{PP} voltage overshoot up to 14V.

In order to meet the rigorous environmental requirements of automotive applications, Intel offers the 27C128 in extended Automotive temperature range. Operational characteristics are guaranteed over the range of -40°C to $+125^{\circ}\text{C}$ ambient.

*HMOS and CHMOS are patented processes of Intel Corporation.

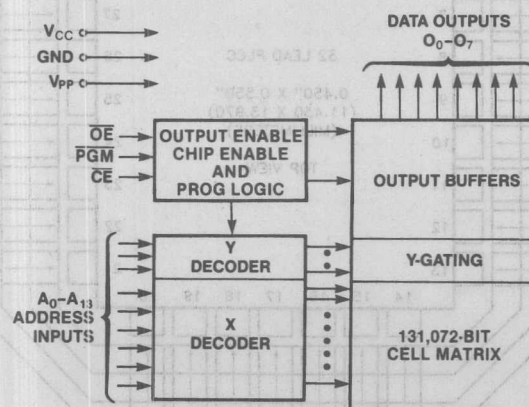
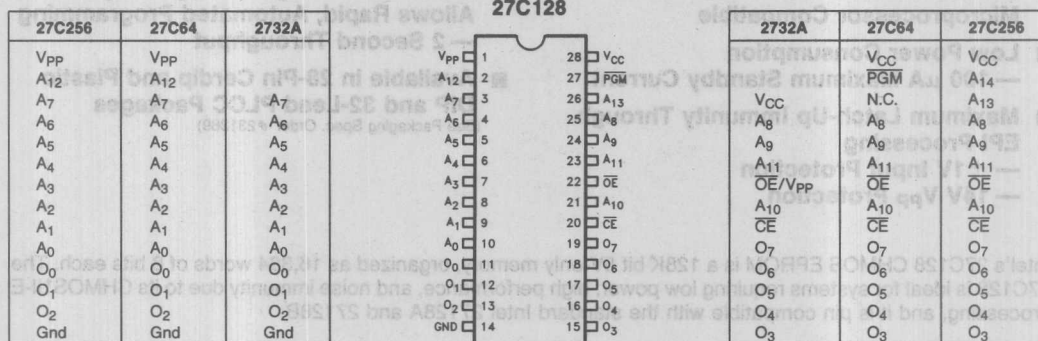


Figure 1. Block Diagram

290140-1

Pin Names

A ₀ -A ₁₃	ADDRESSES
CE	CHIP ENABLE
OE	OUTPUT ENABLE
O ₀ -O ₇	OUTPUTS
PGM	PROGRAM
N.C.	No Internal Connect
D.U.	Don't Use



NOTE:

Intel "Universal Site"-Compatible EPROM Pin Configurations are Shown in the Blocks Adjacent to the 27C128 Pins.

Figure 2. Cerdip(D)/Plastic(P) DIP Pin Configurations

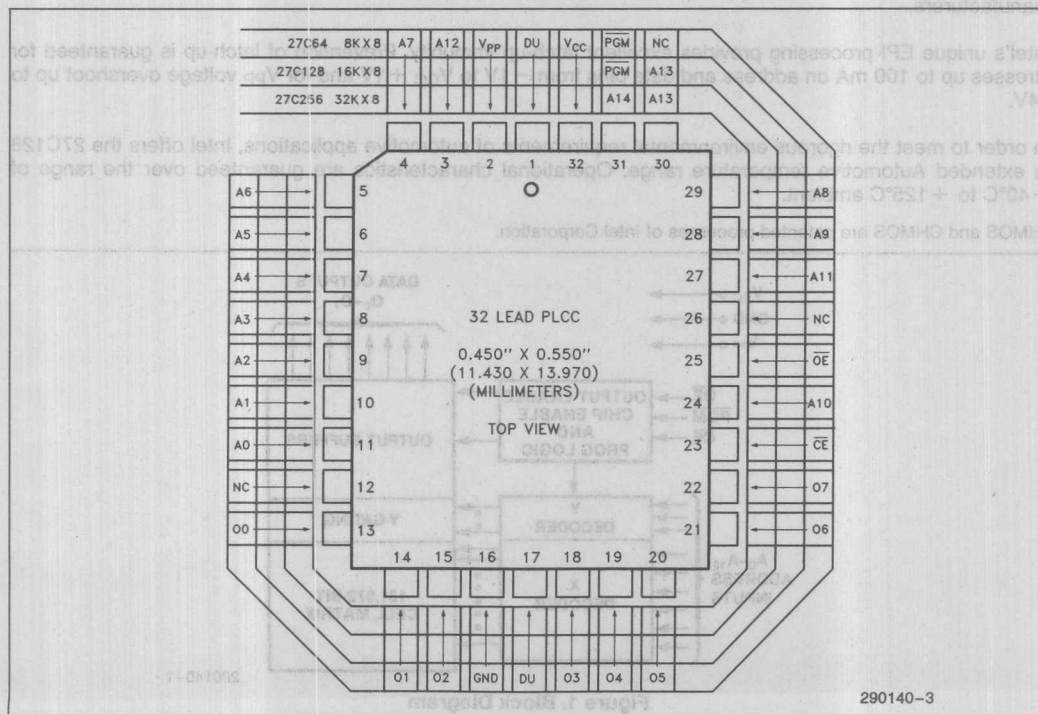


Figure 3. PLCC(N) Lead Configuration

Extended Temperature (Express) EPROMs

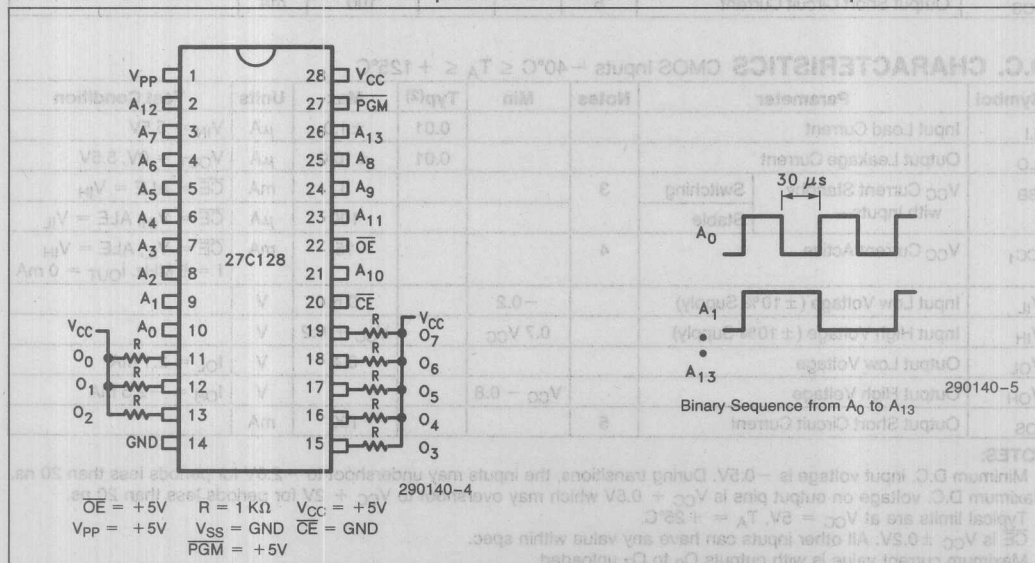
The Intel EXPRESS EPROM family is a series of electrically programmable read only memories which have received additional processing to enhance product characteristics. EXPRESS processing is available for several densities of EPROM, allowing the choice of appropriate memory size to match system applications.

EXPRESS EPROM products are available with 168 ± 8 hour, 125°C dynamic burn-in using Intel's standard bias configuration. This process exceeds or meets most industry specifications of burn-in. The standard EXPRESS EPROM operating temperature range is 0°C to 70°C. Extended operating temperature range (-40°C to +85°C) EXPRESS products are available along with automotive temperature range (-40°C to +125°C) products.

READ OPERATION

D.C. CHARACTERISTICS

Electrical Parameters of EXPRESS EPROM products are identical to standard EPROM parameters



Burn-in Bias and Timing Diagrams

EXPRESS EPROM PRODUCT FAMILY

PRODUCT DEFINITIONS

Type	Operating Temperature (°C)	Burn-in 125°C (hr)
Q	0 to +70	168 ± 8
T	-40 to +85	NONE
L	-40 to +85	168 ± 8
A	-40 to +125	NONE
B	-40 to +125	168 ± 8

AUTOMOTIVE AND EXPRESS OPTIONS

27C128 Versions

Speed Versions	Cerdip	PLCC	Plastic DIP
-150V05	T, L, Q		
-170V05	T, L, Q	T	T
-200V05	T, L, Q, A	T, A	T, A
-250V05	T, L, Q, A	T, A	T, A

Packaging Options

ABSOLUTE MAXIMUM RATINGS*

Operating Temperature During	
Read	−40°C to +125°C
Temperature Under Bias	−40°C to +125°C
Storage Temperature	−65°C to +150°C
Voltage on any Pin with	
Respect to Ground	−2V to +7V(1)
Voltage on A ₉ with	
Respect to Ground	−2V to +13.5V(1)
V _{PP} Supply Voltage with Respect to Ground	
During Programming	−2V to +14.0V(1)
V _{CC} Supply Voltage with	
Respect to Ground	−2V to +7.0V(1)

Maximum Junction Temperature (T_J) 130°C
Maximum Thermal Resistance Junction-to-Ambient (θ_{JA})

Cerdip 43°C/W
Plastic 93°C/W

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

READ OPERATION

D.C. CHARACTERISTICS TTL and NMOS Inputs −40°C ≤ T_A ≤ +125°C

Symbol	Parameter	Notes	Min	Typ(2)	Max	Units	Test Condition
I _{LI}	Input Load Current			0.01	±1.0	μA	V _{IN} = 0V, 5.5V
I _{LO}	Output Leakage Current			0.01	±10	μA	V _{OUT} = 0V, 5.5V
I _{SB}	V _{CC} Current Standby with Inputs—	Switching			10	mA	CE = ALE = V _{IH}
		Stable			1.0	mA	CE = V _{IH} , ALE = V _{IL}
I _{CC1}	V _{CC} Current Active	4			30	mA	CE = V _{IL} , ALE = V _{IH} f = 5 MHz, I _{OUT} = 0 mA
V _{IL}	Input Low Voltage (±10% Supply)	1	−0.5		0.8	V	
V _{IH}	Input High Voltage (±10% Supply)		2.0		V _{CC} + 0.5	V	
V _{OL}	Output Low Voltage				0.45	V	I _{OL} = 2.1 mA
V _{OH}	Output High Voltage		2.4			V	I _{OH} = −400 μA
I _{OS}	Output Short Circuit Current	5			100	mA	

D.C. CHARACTERISTICS CMOS Inputs −40°C ≤ T_A ≤ +125°C

Symbol	Parameter	Notes	Min	Typ(2)	Max	Units	Test Condition
I _{LI}	Input Load Current			0.01	±1.0	μA	V _{IN} = 5.5V
I _{LO}	Output Leakage Current			0.01	±10.0	μA	V _{OUT} = 0V, 5.5V
I _{SB}	V _{CC} Current Standby with Inputs—	Switching	3		6	mA	CE = ALE = V _{IH}
		Stable			100	μA	CE = V _{IH} , ALE = V _{IL}
I _{CC1}	V _{CC} Current Active	4			15	mA	CE = V _{IL} , ALE = V _{IH} f = 5 MHz, I _{OUT} = 0 mA
V _{IL}	Input Low Voltage (±10% Supply)		−0.2		0.8	V	
V _{IH}	Input High Voltage (±10% Supply)		0.7 V _{CC}		V _{CC} + 0.2	V	
V _{OL}	Output Low Voltage				0.4	V	I _{OL} = 2.1 mA
V _{OH}	Output High Voltage		V _{CC} − 0.8			V	I _{OH} = −2.5 mA
I _{OS}	Output Short Circuit Current	5			100	mA	

NOTES:

1. Minimum D.C. input voltage is −0.5V. During transitions, the inputs may undershoot to −2.0V for periods less than 20 ns. Maximum D.C. voltage on output pins is V_{CC} + 0.5V which may overshoot to V_{CC} + 2V for periods less than 20 ns.
2. Typical limits are at V_{CC} = 5V, T_A = +25°C.
3. CE is V_{CC} ±0.2V. All other inputs can have any value within spec.
4. Maximum current value is with outputs O₀ to O₇ unloaded.
5. Output shorted for no more than one second. No more than one output shorted at a time. I_{OS} is sampled but not 100% tested.

READ OPERATION

A.C. CHARACTERISTICS $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$

Versions(3)	$V_{CC} \pm 5\%$	D27C128-150V05	D27C128-170V05 P27C128-170V05 N27C128-170V05	D27C128-200V05 P27C128-200V05 N27C128-200V05	D27C128-250V05 P27C128-250V05 N27C128-250V05	Unit
Symbol	Characteristic	Min	Max	Min	Max	Unit
t_{ACC}	Address to Output Delay		150		170	ns
t_{CE}	\overline{CE} to Output Delay		150		170	ns
t_{OE}	\overline{OE} to Output Delay		75		75	ns
$t_{DF(2)}$	\overline{OE} High to Output High Z		35		55	ns
$t_{OH(2)}$	Output Hold from Addresses, \overline{CE} or \overline{OE} Change-Whichever is First	0		0		ns

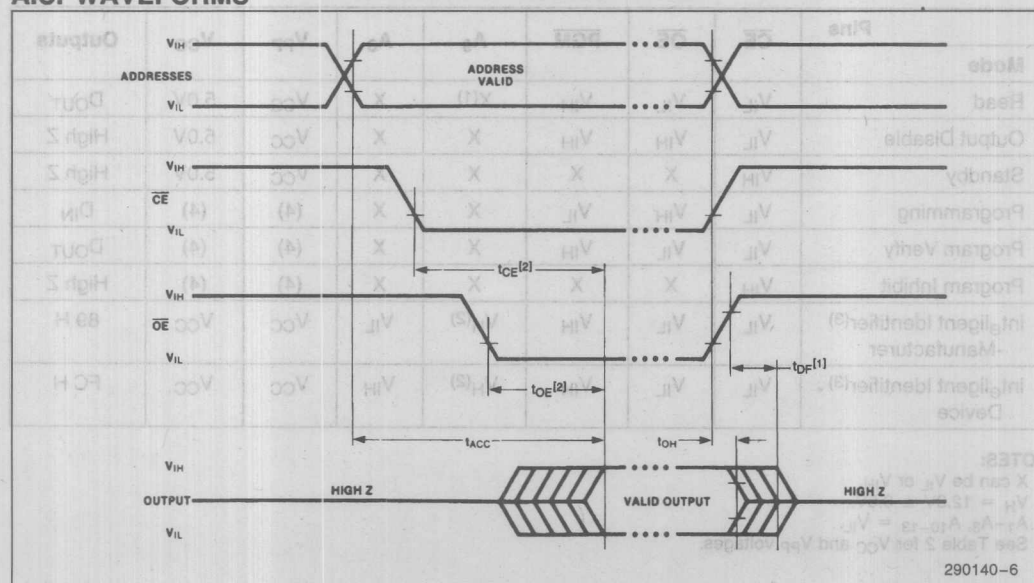
NOTES:

1. A.C. characteristics tested at $V_{IH} = 2.4V$ and $V_{IL} = 0.45V$.
Timing measurements made at $V_{OL} = 0.8V$ and $V_{OH} = 3.5V$.
2. Guaranteed and sampled.
3. Part Number Prefixes: D = CerDip; P = Plastic DIP;
N = PLCC.

A.C. CONDITIONS OF TEST

Input Rise and Fall Times (10% to 90%) 10 ns
 Input Pulse Levels 0.45V to 2.4V
 Input Timing Reference Level 0.8V and 2.0V
 Output Timing Reference Level 0.8V and 2.4V

A.C. WAVEFORMS



NOTES:

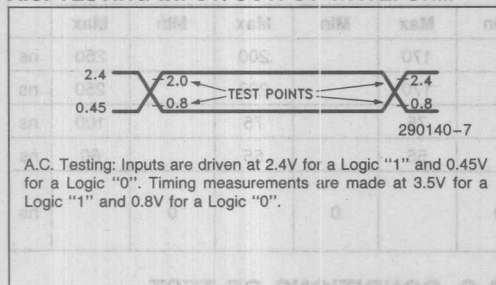
1. This parameter is only sampled and is not 100% tested.
2. \overline{OE} may be delayed up to $t_{CE} - t_{OE}$ after the falling edge of \overline{CE} without impact on t_{CE} .

Symbol	Parameter	Max	Unit	Conditions
C_{IN}	Address/Control Capacitance	6	pF	$V_{IN} = 0V$
C_{OUT}	Output Capacitance	12	pF	$V_{OUT} = 0V$

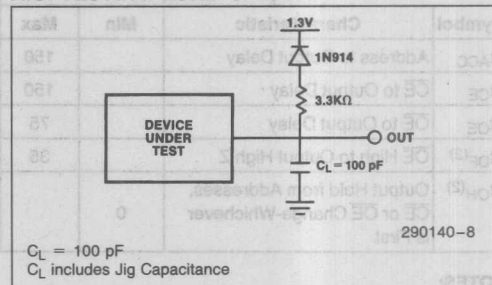
NOTE:

1. Sampled. Not 100% tested.

A.C. TESTING INPUT/OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT



DEVICE OPERATION

The modes of operation of the 27C128 are listed in Table 1. A single 5V power supply is required in the read mode. All inputs are TTL levels except for V_{PP} and 12V on A_9 for intelligent Identifier mode.

Table 1. Mode Selection for 27C128

Pins	\overline{CE}	\overline{OE}	\overline{PGM}	A_9	A_0	V_{PP}	V_{CC}	Outputs
Read	V_{IL}	V_{IL}	V_{IH}	X(1)	X	V_{CC}	5.0V	D_{OUT}
Output Disable	V_{IL}	V_{IH}	V_{IH}	X	X	V_{CC}	5.0V	High Z
Standby	V_{IH}	X	X	X	X	V_{CC}	5.0V	High Z
Programming	V_{IL}	V_{IH}	V_{IL}	X	X	(4)	(4)	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IH}	X	X	(4)	(4)	D_{OUT}
Program Inhibit	V_{IH}	X	X	X	X	(4)	(4)	High Z
intelligent Identifier(3) -Manufacturer	V_{IL}	V_{IL}	V_{IH}	$V_H^{(2)}$	V_{IL}	V_{CC}	V_{CC}	89 H
intelligent Identifier(3) Device	V_{IL}	V_{IL}	V_{IH}	$V_H^{(2)}$	V_{IH}	V_{CC}	V_{CC}	FC H

NOTES:

1. X can be V_{IL} or V_{IH} .
2. $V_H = 12.0V \pm 0.5V$.
3. $A_1-A_8, A_{10-13} = V_{IL}$.
4. See Table 2 for V_{CC} and V_{PP} voltages.

READ MODE

The 27C128 has two control functions, both of which must be logically active in order to obtain data at the outputs. Chip Enable (\overline{CE}) is the power control and should be used for device selection. Output Enable (\overline{OE}) is the output control and should be used to gate data from the output pins, independent of device selection. Assuming that addresses are stable, the address access time (t_{ACC}) is equal to the delay from \overline{CE} to output (t_{CE}). Data is available at the outputs after the delay of t_{OE} from the falling edge of \overline{OE} , assuming that \overline{CE} has been low and addresses have been stable for at least $t_{ACC} - t_{OE}$.

STANDBY MODE

EPRoms can be placed in standby mode which reduces the maximum current of the device by applying a TTL-high signal to the \overline{CE} input. When in standby mode, the outputs are in a high impedance state, independent of the \overline{OE} input.

Two Line Output Control

Because EPRoms are usually used in larger memory arrays, Intel has provided 2 control lines which accommodate this multiple memory connection. The two control lines allow for:

- a) the lowest possible memory power dissipation, and
- b) complete assurance that output bus contention will not occur.

To use these two control lines most efficiently, \overline{CE} should be decoded and used as the primary device selecting function, while \overline{OE} should be made a common connection to all devices in the array and connected to the \overline{READ} line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are active only when data is desired from a particular memory device.

SYSTEM CONSIDERATIONS

The power switching characteristics of EPRoms require careful decoupling of the devices. The supply current, I_{CC} , has three segments that are of interest

to the system designer—the standby current level, the active current level, and the transient current peaks that are produced by the falling and rising edges of Chip Enable. The magnitude of these transient and inductive current peaks is dependent on the output capacitive and inductive loading of the device. The associated transient voltage peaks can be suppressed by complying with Intel's Two-Line Control, and by properly selected decoupling capacitors. It is recommended that a 0.1 μF ceramic capacitor be used on every device between V_{CC} and GND. This should be a high frequency capacitor for low inherent inductance and should be placed as close to the device as possible. In addition, a 4.7 μF bulk electrolytic capacitor should be used between V_{CC} and GND for every eight devices. The bulk capacitor should be located near where the power supply is connected to the array. The purpose of the bulk capacitor is to overcome the voltage droop caused by the inductive effect of PC board-traces.

PROGRAMMING MODES

Caution: Exceeding 14V on V_{PP} will permanently damage the device.

Initially, and after each erasure, all bits of the EPROM are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The device is in the programming mode when V_{PP} is raised to its programming voltage (See Table 2) and \overline{CE} and \overline{PGM} are both at TTL low and $\overline{OE} = V_{IH}$. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

Program Inhibit

Programming of multiple EPRoms in parallel with different data is easily accomplished by using the Program Inhibit mode. A high-level \overline{CE} or \overline{PGM} input inhibits the other devices from being programmed. Except for \overline{CE} , all like inputs (including \overline{OE}) of the parallel EPRoms may be common. A TTL low-level pulse applied to the \overline{PGM} input with V_{PP} at its programming voltage and $\overline{CE} = V_{IL}$ will program the selected device.

intelligent Identifier™ Mode

The intelligent Identifier Mode allows the reading out of a binary code from an EPROM that will identify its manufacturer and type. This mode is intended for use by programming equipment for the purpose of automatically matching the device to be programmed with its corresponding programming algorithm. This mode is functional in the $25^{\circ}\text{C} \pm 5^{\circ}\text{C}$ ambient temperature range that is required when programming the device.

To activate this mode, the programming equipment must force 11.5V to 12.5V on address line A9 of the EPROM. Two identifier bytes may then be sequenced from the device outputs by toggling address line A0 from V_{IL} to V_{IH} . All other address lines must be held at V_{IL} during the intelligent Identifier Mode.

Byte 0 ($A0 = V_{IL}$) represents the manufacturer code and byte 1 ($A0 = V_{IH}$) the device identifier code. These two identifier bytes are given in Table 1.

ERASURE CHARACTERISTICS (FOR CERDIP EPROMS)

The erasure characteristics are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å range. Data shows that constant exposure to room level fluorescent lighting could erase

the EPROM in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the device is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the window to prevent unintentional erasure.

The recommended erasure procedure is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity x exposure time) for erasure should be a minimum of 15 Wsec/cm². The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 $\mu\text{W}/\text{cm}^2$ power rating. The EPROM should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated dose an EPROM can be exposed to without damage is 7258 Wsec/cm² (1 week @ 12000 $\mu\text{W}/\text{cm}^2$). Exposure of the device to high intensity UV light for longer periods may cause permanent damage.

HIGH RELIABILITY CHMOS

Special EPI processing techniques have enabled Intel to build CHMOS with features adding to system reliability. These include input/output protection to latch-up. Each of the data and address pins will not latch-up with currents up to 100 mA and voltages from -1V to $V_{CC} + 1\text{V}$.

Additionally, the V_{PP} (programming) pin is designed to resist latch-up to the 14V maximum device limit.

SYSTEM CONSIDERATIONS

The power switching characteristics of EPROMs require careful decoupling of the devices. The supply current, I_{CC} , has three segments that are of interest

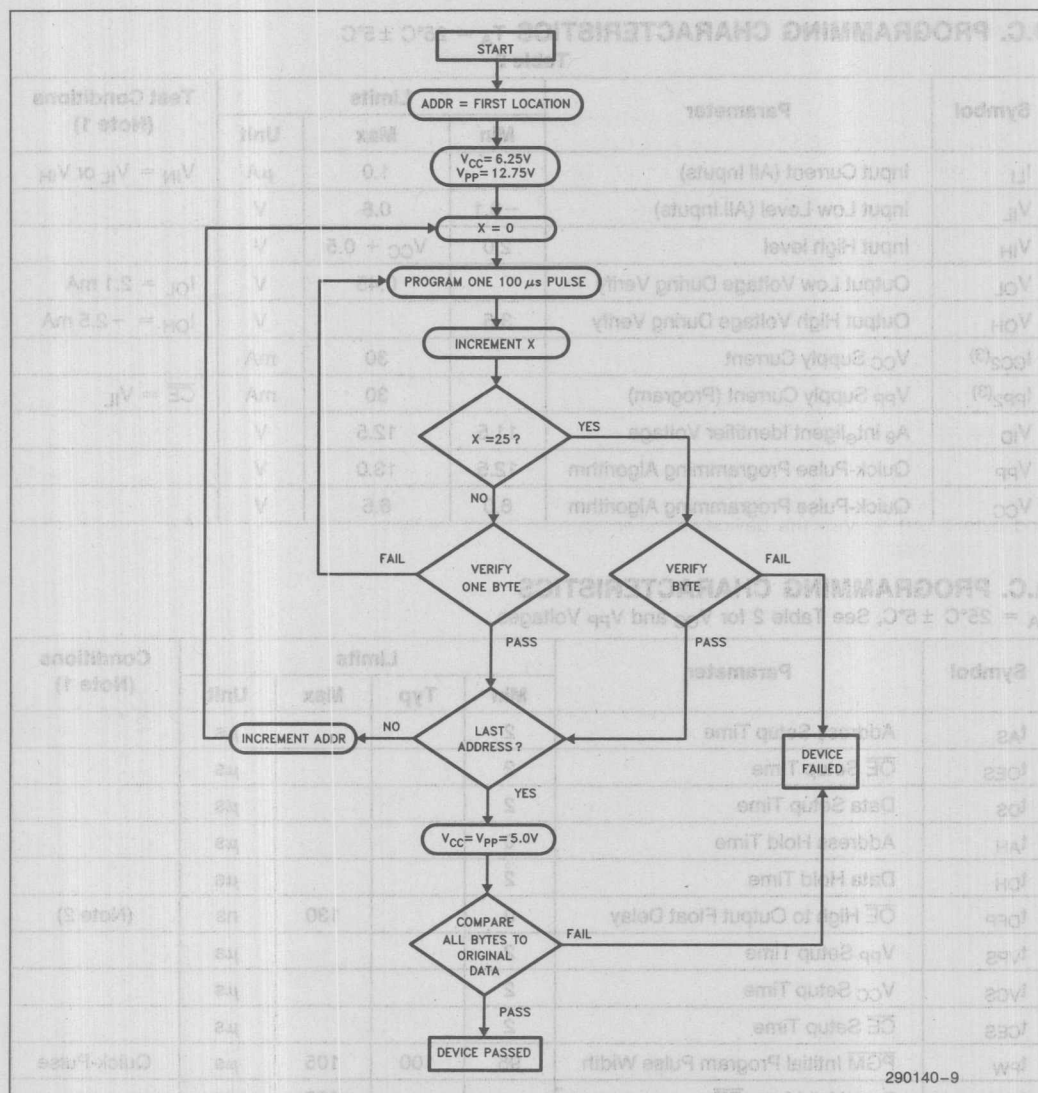


Figure 5. Quick-Pulse Programming™ Algorithm

Quick-Pulse Programming™ Algorithm

Intel's 27C128 EPROM is programmed using the Quick-Pulse Programming Algorithm, developed by Intel to substantially reduce the throughput time in the production environment. This algorithm allows the device to be programmed in under two seconds, almost a hundred fold improvement over previous algorithms. Actual programming time is a function of the PROM programmer being used.

The Quick-Pulse Programming Algorithm uses initial pulses of 100 microseconds followed by a byte veri-

fication to determine when the address byte has been successfully programmed. Up to 25 100 μs pulses per byte are provided before a failure is recognized. A flowchart of the Quick-Pulse Programming Algorithm is shown in Figure 5.

For the Quick Pulse Programming Algorithm, the entire sequence of programming pulses and byte verifications is performed at $V_{CC} = 6.25V$ and $V_{pp} = 12.75V$. When programming of the EPROM has been completed, all bytes should be compared to the original data with $V_{CC} = V_{pp} = 5.0V$.

Table 2

Symbol	Parameter	Limits			Test Conditions (Note 1)
		Min	Max	Unit	
I_{LI}	Input Current (All Inputs)		1.0	μA	$V_{IN} = V_{IL} \text{ or } V_{IH}$
V_{IL}	Input Low Level (All Inputs)	-0.1	0.8	V	
V_{IH}	Input High level	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage During Verify		0.45	V	$I_{OL} = 2.1 \text{ mA}$
V_{OH}	Output High Voltage During Verify	3.5		V	$I_{OH} = -2.5 \text{ mA}$
$I_{CC2}^{(3)}$	V_{CC} Supply Current		30	mA	
$I_{PP2}^{(3)}$	V_{PP} Supply Current (Program)		30	mA	$\overline{CE} = V_{IL}$
V_{ID}	A_9 intelligent Identifier Voltage	11.5	12.5	V	
V_{PP}	Quick-Pulse Programming Algorithm	12.5	13.0	V	
V_{CC}	Quick-Pulse Programming Algorithm	6.0	6.5	V	

A.C. PROGRAMMING CHARACTERISTICS

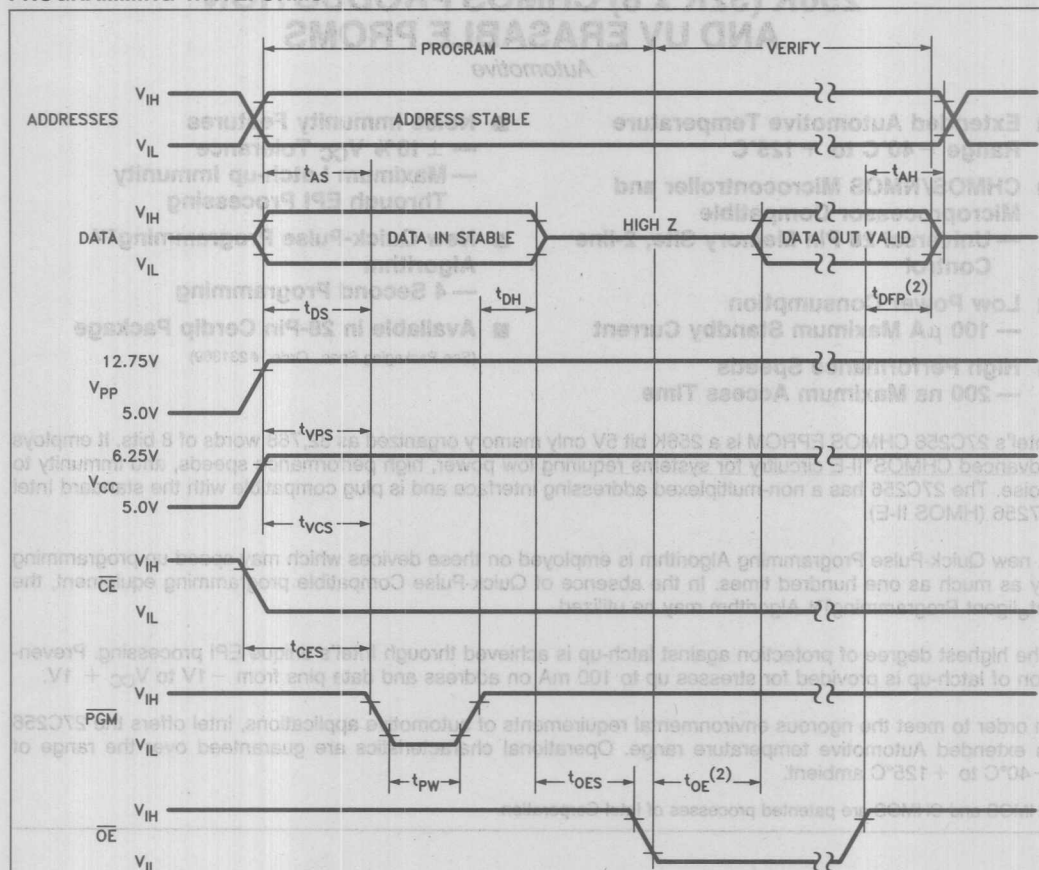
$T_A = 25^\circ C \pm 5^\circ C$, See Table 2 for V_{CC} and V_{PP} Voltages

Symbol	Parameter	Limits				Conditions (Note 1)
		Min	Typ	Max	Unit	
t_{AS}	Address Setup Time	2			μs	
t_{OES}	\overline{OE} Setup Time	2			μs	
t_{DS}	Data Setup Time	2			μs	
t_{AH}	Address Hold Time	0			μs	
t_{DH}	Data Hold Time	2			μs	
t_{DFP}	\overline{OE} High to Output Float Delay	0		130	ns	(Note 2)
t_{VPS}	V_{PP} Setup Time	2			μs	
t_{VCS}	V_{CC} Setup Time	2			μs	
t_{CES}	\overline{CE} Setup Time	2			μs	
t_{PW}	PGM Initial Program Pulse Width	95	100	105	μs	Quick-Pulse
t_{OE}	Data Valid from \overline{OE}			150	ns	

NOTE 3:

- V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP} .
- This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.
- The maximum current value is with outputs O_0 to O_7 Unloaded.

PROGRAMMING WAVEFORMS



290140-10

NOTES:

1. The Input Timing Reference Level is 0.8V for V_{IL} and 2V for a V_{IH} .
2. t_{OE} and t_{DFP} are characteristics of the device but must be accommodated by the programmer.
3. When programming the 27C128, a 0.1 μF capacitor is required across V_{PP} and ground to suppress spurious voltage transients which can damage the device.

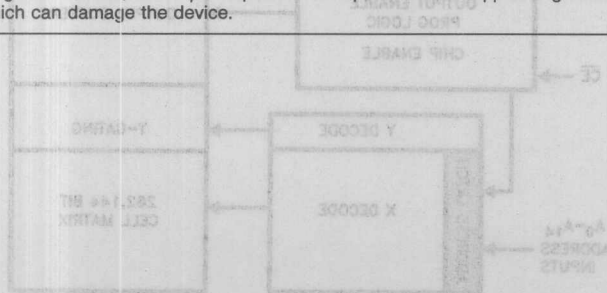


Figure 1: Block Diagram



27C256

256K (32K x 8) CHMOS PRODUCTION AND UV ERASABLE PROMS

Automotive

- **Extended Automotive Temperature Range** — -40°C to $+125^{\circ}\text{C}$
- **CHMOS/NMOS Microcontroller and Microprocessor Compatible**
 - Universal 28 Pin Memory Site, 2-line Control
- **Low Power Consumption**
 - $100\ \mu\text{A}$ Maximum Standby Current
- **High Performance Speeds**
 - $200\ \text{ns}$ Maximum Access Time
- **Noise Immunity Features**
 - $\pm 10\%$ V_{CC} Tolerance
 - Maximum Latch-up Immunity Through EPI Processing
- **New Quick-Pulse Programming™ Algorithm**
 - 4 Second Programming
- **Available in 28-Pin Cerdip Package**
(See Packaging Spec., Order #231369)

Intel's 27C256 CHMOS EPROM is a 256K bit 5V only memory organized as 32,768 words of 8 bits. It employs advanced CHMOS*II-E circuitry for systems requiring low power, high performance speeds, and immunity to noise. The 27C256 has a non-multiplexed addressing interface and is plug compatible with the standard Intel 27256 (HMOS II-E).

A new Quick-Pulse Programming Algorithm is employed on these devices which may speed up programming by as much as one hundred times. In the absence of Quick-Pulse Compatible programming equipment, the intelligent Programming™ Algorithm may be utilized.

The highest degree of protection against latch-up is achieved through Intel's unique EPI processing. Prevention of latch-up is provided for stresses up to 100 mA on address and data pins from -1V to $V_{\text{CC}} + 1\text{V}$.

In order to meet the rigorous environmental requirements of automotive applications, Intel offers the 27C256 in extended Automotive temperature range. Operational characteristics are guaranteed over the range of -40°C to $+125^{\circ}\text{C}$ ambient.

*HMOS and CHMOS are patented processes of Intel Corporation.

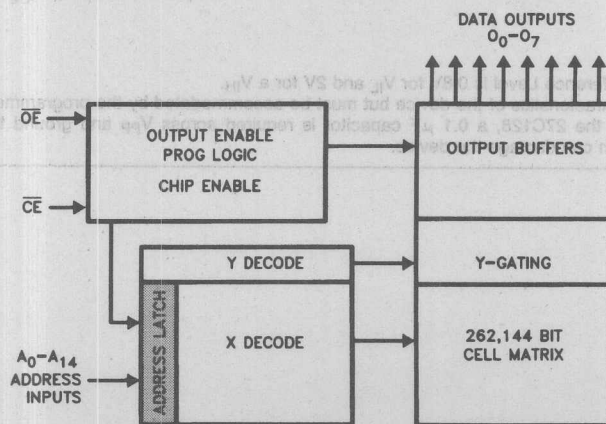


Figure 1. Block Diagram

290120-1

Pin Names

A ₀ -A ₁₄	ADDRESSES
O ₀ -O ₇	OUTPUTS
OE	OUTPUT ENABLE
CE	CHIP ENABLE
N.C.	NO CONNECT
D.U.	DON'T USE

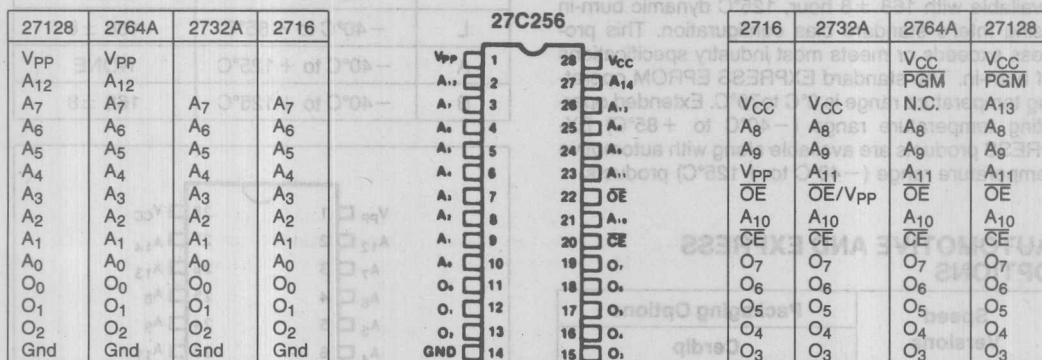


Figure 2. Pin Configuration

NOTE:

Intel "Universal Site" -Compatible EPROM Pin Configurations are Shown in the Blocks Adjacent.

EXTENDED TEMPERATURE (EXPRESS) EPROMs

The Intel EXPRESS EPROM family is a series of electrically programmable read only memories which have received additional processing to enhance product characteristics. EXPRESS processing is available for several densities of EPROM, allowing the choice of appropriate memory size to match system applications. EXPRESS EPROM products are available with 168 ± 8 hour, 125°C dynamic burn-in using Intel's standard bias configuration. This process exceeds or meets most industry specifications of burn-in. The standard EXPRESS EPROM operating temperature range is 0°C to 70°C . Extended operating temperature range (-40°C to $+85^\circ\text{C}$) EXPRESS products are available along with automotive temperature range (-40°C to $+125^\circ\text{C}$) products.

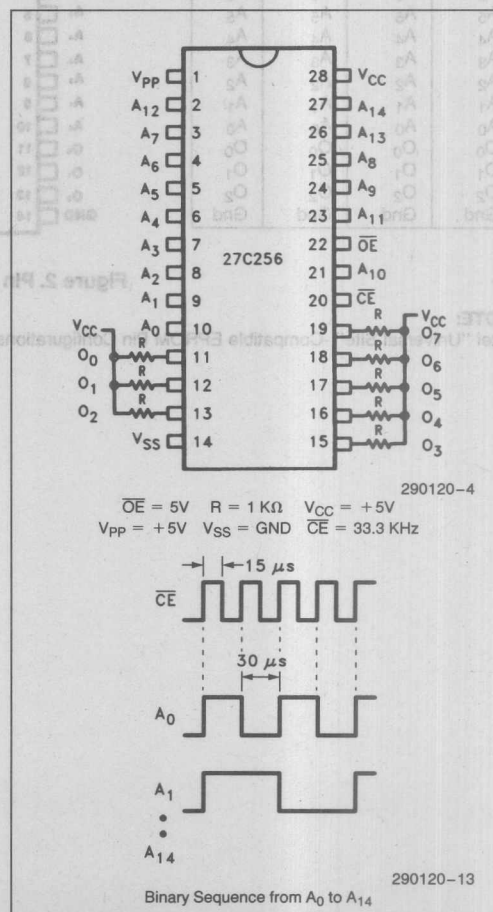
AUTOMOTIVE AND EXPRESS OPTIONS

Speed Versions	Packaging Options
	Cerdip
-2	T, L, Q, A
-20	T, L, Q, A
STD	T, L, Q, A
-25	T, L, Q, A
-3	T, L, Q, A
-30	T, L, Q, A

AUTOMOTIVE AND EXPRESS EPROM PRODUCT FAMILY

PRODUCT DEFINITIONS

Type	Operating Temperature ($^\circ\text{C}$)	Burn-In 125°C (hr)
Q	0°C to $+70^\circ\text{C}$	168 ± 8
T	-40°C to $+85^\circ\text{C}$	NONE
L	-40°C to $+85^\circ\text{C}$	168 ± 8
A	-40°C to $+125^\circ\text{C}$	NONE
B	-40°C to $+125^\circ\text{C}$	168 ± 8



Burn-In Bias and Timing Diagrams

ABSOLUTE MAXIMUM RATINGS*

Operating Temperature During Read	−40°C to +125°C
Temperature Under Bias	−40°C to +125°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to Ground	−2V to +7V(1)
Voltage on A ₉ with Respect to Ground	−2V to +13.5V(1)
V _{PP} supply Voltage with Respect to Ground during programming	−2V to +14.0V(1)
V _{CC} Supply Voltage with Respect to Ground	−2V to +7.0V(1)
Maximum Junction Temperature (T _J)	140°C

Maximum Thermal Resistance
Junction to Ambient (θ_{JA}):

Cerdip	36°C/W
PLCC	55°C/W

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

READ OPERATION

D.C. CHARACTERISTICS: 27C256 −40°C ≤ T_A ≤ +125°C

Symbol	Parameter	Notes	Min	Typ ⁽²⁾	Max	Units	Test Condition
I _{LI}	Input Load Current			0.01	±1.0	μA	V _{IN} = 0V, 5.5V
I _{LO}	Output Leakage Current			0.01	±10	μA	V _{OUT} = 0V, 5.5V
I _{PP1}	V _{PP} Read Current	4			200	μA	V _{PP} = V _{CC}
I _{SB}	V _{CC} Current Standby	CMOS 3			200	μA	CE = V _{CC}
I _{CC1}	V _{CC} Active Current (mA)	TTL			30		OE = CE = V _{IL}
	V _{CC} Active Current at High Temperature (mA)	TTL			30		OE = CE = V _{IL} V _{PP} = V _{CC} T _{Ambient} = +125°C
V _{IL}	Input Low Voltage (±10% Supply) (TTL)		−0.5		0.8	V	V _{PP} = V _{CC}
	Input Low Voltage (CMOS)		−0.2		0.2		
V _{IH}	Input High Voltage (±10% Supply) (TTL)		2.0		V _{CC} + 0.5	V	V _{PP} = V _{CC}
	Input High Voltage (CMOS)		V _{CC} − 0.2		V _{CC} + 0.2		
V _{OL}	Output Low Voltage				0.45	V	I _{OL} = 2.1 mA
V _{OH}	Output High Voltage		3.5			V	I _{OH} = −2.5 mA
I _{OS}	Output Short Circuit Current	5			100	mA	
V _{PP}	V _{PP} Read Voltage	6	V _{CC} − 0.7		V _{CC}	V	

NOTES:

1. Minimum D.C. input voltage is −0.5V. During transitions, the inputs may undershoot to −2.0V for periods less than 20 ns. Maximum D.C. voltage on output pins is V_{CC} + 0.5V which may overshoot to V_{CC} + 2V for periods less than 20 ns.
2. Typical limits are at V_{CC} = 5V, T_A = +25°C.
3. CE is V_{CC} ±0.2V. All other inputs can have any value within spec.

4. Maximum Active power usage is the sum I_{PP} + I_{CC}. The maximum current value is with outputs O₀ to O₇ unloaded.
5. Output shorted for no more than one second. No more than one output shorted at a time. I_{OS} is sampled but not 100% tested.
6. V_{PP} may be one diode voltage drop below V_{CC}. It may be connected directly to V_{CC}. Also, V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
7. V_{IL}, V_{IH} levels at TTL inputs.

READ OPERATION

A.C. CHARACTERISTICS 27C256(1) $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$

Versions(3)		V _{CC} ± 5%		27C256-2		27C256		27C256-3		Unit
		V _{CC} ± 10%		27C256-20		27C256-25		27C256-30		
Symbol	Characteristic	Min	Max	Min	Max	Min	Max	Min	Max	
t _{ACC}	Address to Output Delay		200		250		300			ns
t _{CE}	\overline{CE} to Output Delay		200		250		300			ns
t _{OE}	\overline{OE} to Output Delay		75		100		120			ns
t _{DF} (2)	\overline{OE} High to Output High Z		55		60		75			ns
t _{OH} (2)	Output Hold from Addresses, \overline{CE} or \overline{OE} Change-Whichever is First	0		0		0				ns

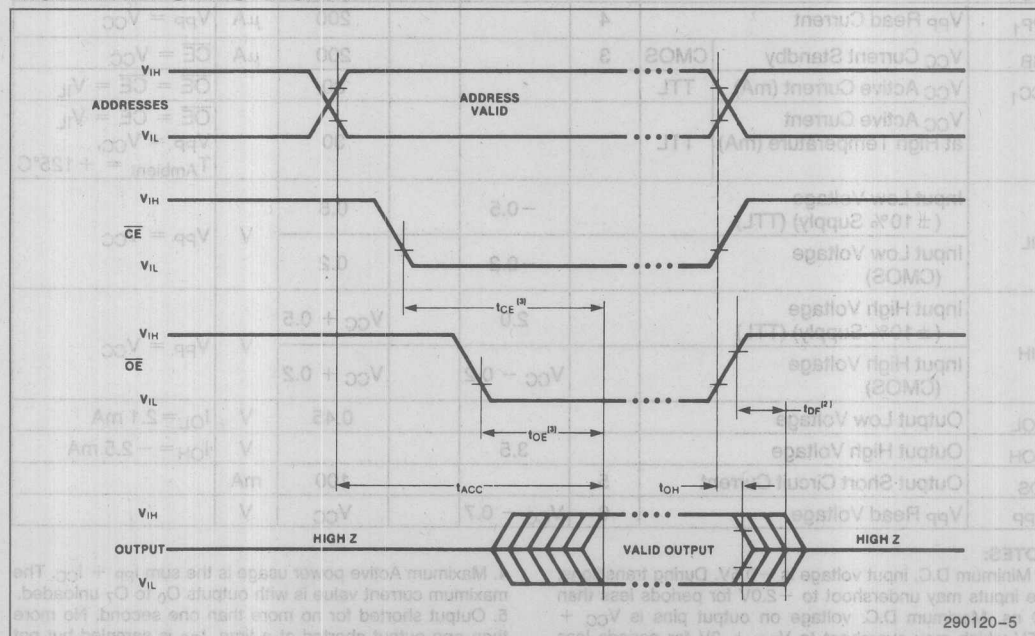
NOTES:

1. A.C. characteristics tested at $V_{IH} = 2.4\text{V}$ and $V_{IL} = 0.45\text{V}$.
Timing measurements made at $V_{OL} = 0.8\text{V}$ and $V_{OH} = 2.0\text{V}$.
2. Guaranteed and sampled.
3. Part Number Prefixes: No Prefix = CERDIP

A.C. CONDITIONS OF TEST

- Input Rise and Fall Times (10% to 90%) 10 ns
 Input Pulse Levels 0.45V to 2.4V
 Input Timing Reference Level 0.8V and 2.0V
 Output Timing Reference Level 0.8V and 2.4V

A.C. WAVEFORMS 27C256



NOTES:

1. Typical values are for $T_A = 25^{\circ}\text{C}$ and nominal supply voltages.
2. This parameter is only sampled and is not 100% tested.
3. \overline{OE} may be delayed up to $t_{CE} - t_{OE}$ after the falling edge of \overline{CE} without impact on t_{CE} .

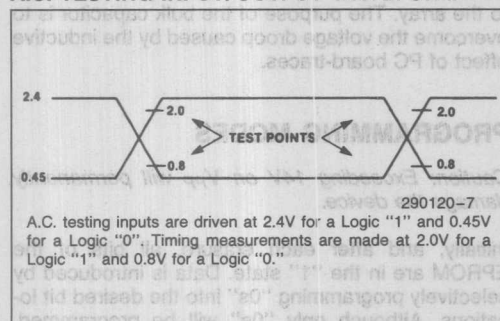
CAPACITANCE(1) $T_A = 25^\circ\text{C}$, $f = 1.0\text{ MHz}$

Symbol	Parameter	Max	Units	Conditions
C_{IN}	Address/control capacitance	6	pF	$V_{IN} = 0\text{V}$
C_{OUT}	Output Capacitance	12	pF	$V_{OUT} = 0\text{V}$

NOTE:

1. Sampled. Not 100% tested.

A.C. TESTING INPUT/OUTPUT WAVEFORM



DEVICE OPERATION

The modes of operation of the 27C256 are listed in Table 1. A single 5V power supply is required in the read mode. All inputs are TTL levels except for V_{PP} and 12V on A_9 for intelligent Identifier™ mode.

Table 1. Mode Selection

Pins	\overline{CE}	\overline{OE}	A_9	A_0	V_{PP}	V_{CC}	Outputs
Read	V_{IL}	V_{IL}	X ⁽¹⁾	X	V_{CC}	5.0V	D_{OUT}
Output Disable	V_{IL}	V_{IH}	X	X	V_{CC}	5.0V	High Z
Standby	V_{IH}	X	X	X	V_{CC}	5.0V	High Z
Programming	V_{IL}	V_{IH}	X	X	(Note 4)	(Note 4)	D_{IN}
Program Verify	V_{IH}	V_{IL}	X	X	(Note 4)	(Note 4)	D_{OUT}
Program Inhibit	V_{IH}	V_{IH}	X	X	(Note 4)	(Note 4)	HIGH Z
intelligent Identifier ⁽³⁾ -Manufacturer	V_{IL}	V_{IL}	V_H ⁽²⁾	V_{IL}	V_{CC}	V_{CC}	89 H
intelligent Identifier ⁽³⁾ -27C256	V_{IL}	V_{IL}	V_H ⁽²⁾	V_{IH}	V_{CC}	V_{CC}	8C H

NOTES:

- X can be V_{IL} or V_{IH} .
- $V_H = 12.0\text{V} \pm 0.5\text{V}$.
- $A_1-A_8, A_{10}-12 = V_{IL}$.
- See Table 2 for V_{CC} and V_{PP} voltages during programming.

Read Mode

The 27C256 has two control functions, both of which must be logically active in order to obtain data at the outputs. Chip Enable (\overline{CE}) is the power control and should be used for device selection. Output enable (\overline{OE}) is the output control and should be used to gate data from the output pins, independent of device selection. Assuming that addresses are stable, the address access time (t_{ACC}) is equal to the delay from \overline{CE} to output (t_{CE}). Data is available at the outputs after a delay of t_{OE} from the falling edge of \overline{OE} , assuming that \overline{CE} has been low and addresses have been stable for at least $t_{ACC}-t_{OE}$.

Standby Mode

The 27C256 has a Standby mode which reduces the maximum V_{CC} current to 100 μA . Both are placed in the Standby mode when \overline{CE} is in the CMOS-high state. When the Standby mode, the outputs are in a high impedance state, independent of the \overline{OE} input.

Two Line Output Control

Because EPROMs are usually used in larger memory arrays, Intel has provided 2 control lines which accommodate this multiple memory connection. The two control lines allow for:

- the lowest possible memory power dissipation, and
- complete assurance that output bus contention will not occur.

To use these two control lines most efficiently, \overline{CE} should be decoded and used as the primary device selecting function, while \overline{OE} should be made a common connection to all devices in the array and connected to the \overline{READ} line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are active only when data is desired from a particular memory device.

SYSTEM CONSIDERATIONS

The power switching characteristics of EPROMs require careful decoupling of the devices. The supply current, I_{CC} , has three segments that are of interest to the system designer—the standby current level, the active current level, and the transient current peaks that are produced by the falling and rising edges of Chip Enable. The magnitude of these transient current peaks is dependent on the output capacitive and inductive loading of the device. The as-

sociated transient voltage peaks can be suppressed by complying with Intel's Two-Line Control, and by properly selected decoupling capacitors. It is recommended that a 0.1 μF ceramic capacitor be used on every device between V_{CC} and GND. This should be a high frequency capacitor for low inherent inductance and should be placed as close to the device as possible. In addition, a 4.7 μF bulk electrolytic capacitor should be used between V_{CC} and GND for every eight devices. The bulk capacitor should be located near where the power supply is connected to the array. The purpose of the bulk capacitor is to overcome the voltage droop caused by the inductive effect of PC board-traces.

PROGRAMMING MODES

Caution: Exceeding 14V on V_{PP} will permanently damage the device.

Initially, and after each erasure, all bits of the EPROM are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The device is in the programming mode when V_{PP} is raised to its programming voltage (See Table 2) and \overline{CE} is pulsed to TTL low and $\overline{OE} = V_{IH}$. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

Program Inhibit

Programming of multiple EPROMs in parallel with different data is easily accomplished by using the Program Inhibit mode. A high-level \overline{CE} input inhibits the other devices from being programmed.

Except for \overline{CE} and \overline{OE} all like inputs of the parallel EPROMs may be common. A TTL low-level pulse applied to the \overline{CE} input with V_{PP} at its programming voltage will program the selected device.

Program Verify

A verify should be performed on the programmed bits to determine that they have been correctly programmed. The verify is performed with \overline{OE} at V_{IL} and \overline{CE} at V_{IH} , and V_{PP} and V_{CC} at their programming voltages. Data should be verified a minimum of t_{OE} after the falling edge of \overline{OE} .

Optional Program Verify

All 27C256s with $V_{PP} = 12.75V$ (12.5V intelligent programming) and $\overline{OE} = V_{IL}$ will present data on the bus independent of the \overline{CE} state. The optional verify may be used in place of the verify mode to allow parallel programming where several devices share a common bus. It is performed with \overline{OE} at V_{IL} , $\overline{CE} = V_{IL}$ (as opposed to the standard verify which has \overline{CE} at V_{IH}), and $V_{PP} = V_{CC} = 6.25V$ (6.0V intelligent programming). The outputs will then tri-state according to the signals presented to \overline{OE} and \overline{CE} . With V_{PP} lowered to V_{CC} ($= 6.25V/6.0V$ —See Table 2), the normal read mode may be used to execute a program verify.

intelligent Identifier™ Mode

The intelligent Identifier Mode allows the reading out of a binary code from an EPROM that will identify its manufacturer and type. This mode is intended for use by programming equipment for the purpose of automatically matching the device to be programmed with its corresponding programming algorithm. This mode is functional in the $25^{\circ}C \pm 5^{\circ}C$ ambient temperature range that is required when programming the device.

To activate this mode, the programming equipment must force 11.5V to 12.5V on address line A_9 of the EPROM. Two identifier bytes may then be sequenced from the device outputs by toggling address line A_0 from V_{IL} to V_{IH} . All other address lines must be held at V_{IL} during the intelligent Identifier Mode.

Byte 0 ($A_0 = V_{IL}$) represents the manufacturer code and byte 1 ($A_0 = V_{IH}$) the device identifier code. These two identifier bytes are given in Table 1.

INTEL EPROM PROGRAMMING SUPPORT TOOLS

Intel offers a full line of EPROM Programmers providing state-of-the-art programming for Intel programmable devices. The modular architecture of Intel's EPROM programmers allows you to add new support as it becomes available, with very low cost add-ons. For example, even the earliest users of the iUP-FAST 27/K module may take advantage of Intel's new Quick-Pulse Programming Algorithm, the fastest in the industry.

Intel EPROM programmers may be controlled from a host computer using Intel's PROM Programming software (iPPS). iPPS makes programming easy for

a growing list of industry standard hosts, including the IBM PC, XT, AT, and PC DOS compatibles, Inteltec Development Systems, Intel's iPDS Personal Development Systems, and the Intel Network Development System (iNDS-II). Stand-alone operation is also available, including device previewing, editing, programming, and download of programming data from any source over an RS232C port.

For further details consult the EPROM Programming section of the Development Systems Handbook.

ERASURE CHARACTERISTICS (FOR Cerdip EPROMS)

The erasure characteristics are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms (\AA). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000 \AA range. Data shows that constant exposure to room level fluorescent lighting could erase the EPROM in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the device is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the window to prevent unintentional erasure.

The recommended erasure procedure is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (\AA). The integrated dose (i.e., UV intensity \times exposure time) for erasure should be a minimum of 15 Wsec/cm². The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 $\mu W/cm^2$ power rating. The EPROM should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated dose an EPROM can be exposed to without damage is 7258 Wsec/cm² (1 week @ 12000 $\mu W/cm^2$). Exposure of the device to high intensity UV light for longer periods may cause permanent damage.

CHMOS NOISE CHARACTERISTICS

Special EPI processing techniques have enabled Intel to build CHMOS with features adding to system reliability. These include input/output protection to latch-up. Each of the data and address pins will not latch-up with currents up to 100 mA and voltages from $-1V$ to $V_{CC} + 1V$.

Additionally, the V_{PP} (programming) pin is designed to resist latch-up to the 14V maximum device limit.

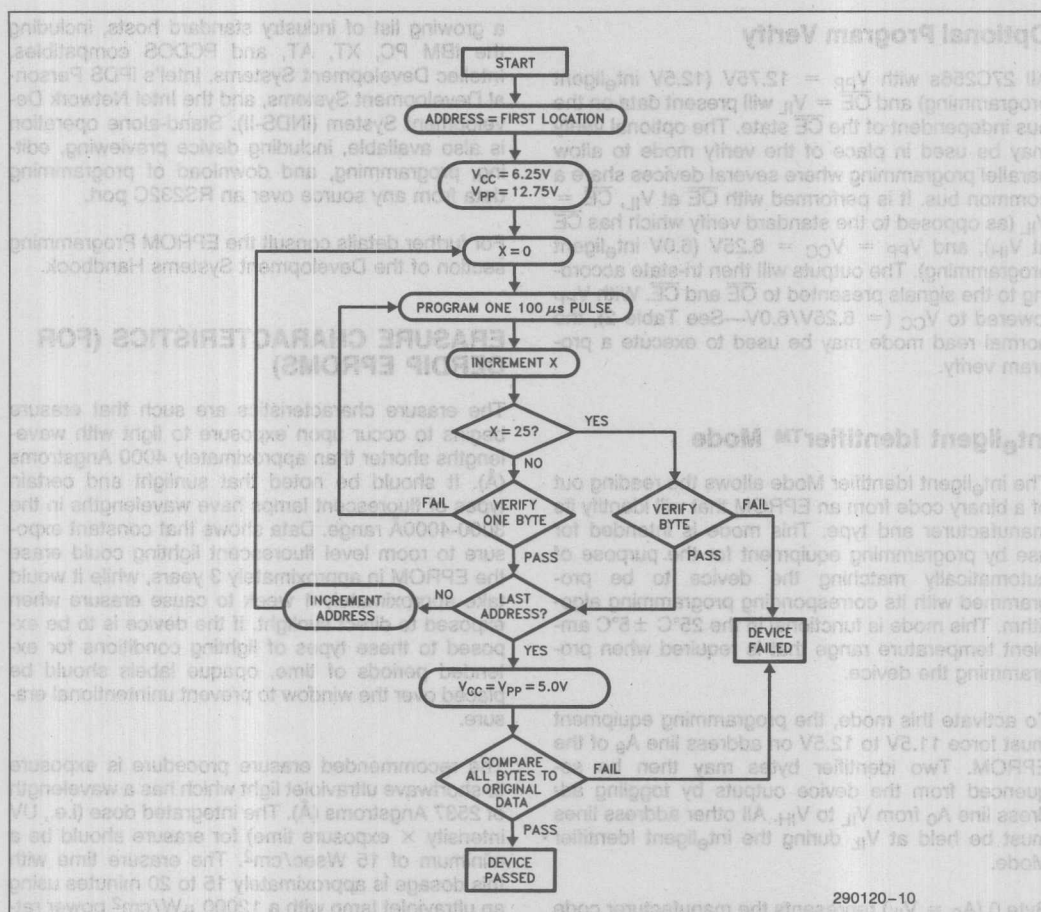


Figure 5. Quick-Pulse Programming™ Algorithm

Quick-Pulse Programming™ Algorithm

Intel's 27C256 EPROMs can now be programmed using the Quick-Pulse Programming Algorithm, developed by Intel to substantially reduce the throughput time in the production programming environment. This algorithm allows 27C256s to be programmed in under four seconds, almost a hundred fold improvement over previous algorithms. Actual programming time is a function of the PROM programmer being used.

The Quick-Pulse Programming Algorithm uses initial pulses of 100 microseconds followed by a byte verification to determine when the address byte has been successfully programmed. Up to 25 100 μ s

pulses per byte are provided before a failure is recognized. A flowchart of the Quick-Pulse Programming Algorithm is shown in Figure 5.

For the Quick-Pulse Programming Algorithm, the entire sequence of programming pulses and byte verifications is performed at $V_{CC} = 6.25V$ and $V_{PP} = 12.75V$. When programming of the EPROM has been completed, all bytes should be compared to the original data with $V_{CC} = V_{PP} = 5.0V$.

In addition to the Quick-Pulse Programming Algorithm, the 27C256 is also compatible with Intel's Intelligent Programming Algorithm.

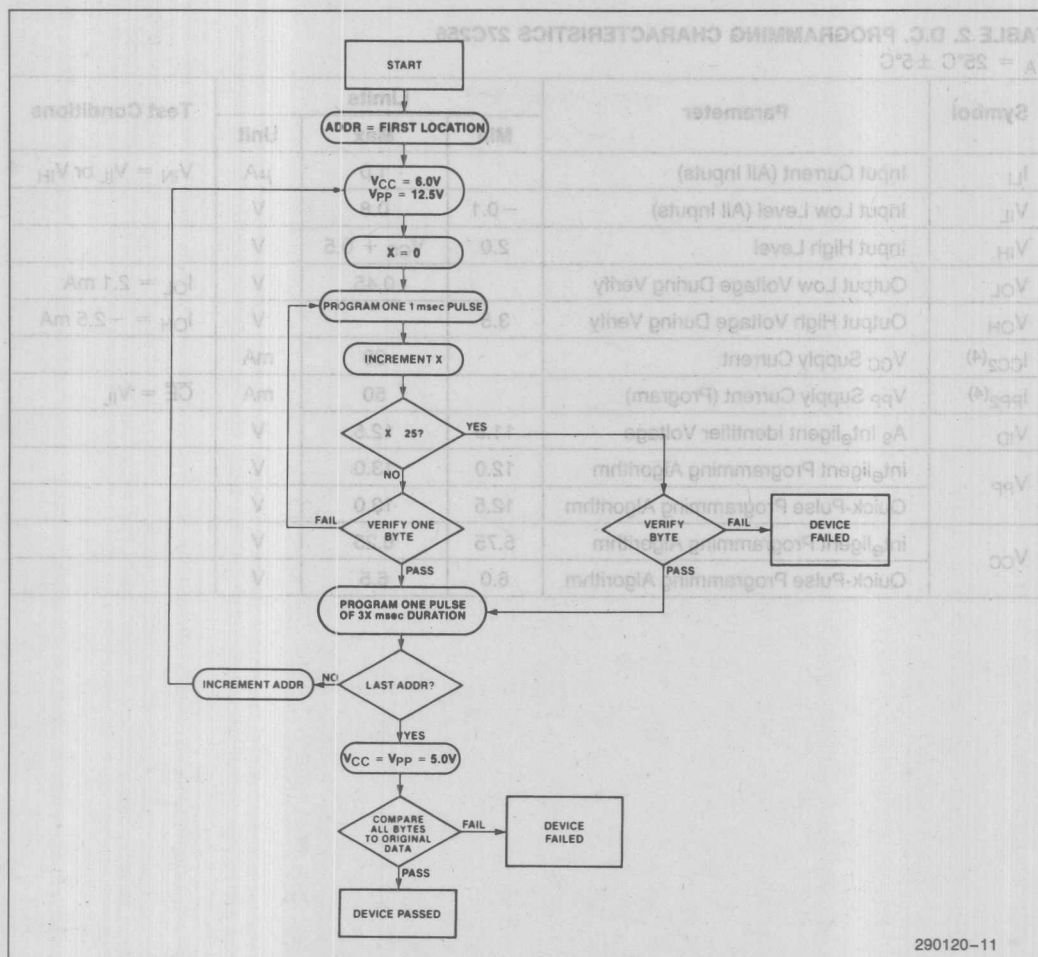


Figure 6. intelligent Programming™ Flowchart

intelligent Programming™ Algorithm

The intelligent Programming Algorithm has been a standard in the industry for the past few years. A flow-chart of the intelligent Programming Algorithm is shown in Figure 6.

The intelligent Programming Algorithm utilizes two different pulse types: initial and overprogram. The duration of the initial pulse(s) is one millisecond, which will then be followed by a larger overprogram pulse of length 3X msec. X is an iteration counter

and is equal to the number of the initial one millisecond pulses applied to a particular location, before a correct verify occurs. Up to 25 one-millisecond pulses per byte are provided for before the overprogram pulse is applied.

The entire sequence of program pulses and byte verifications is performed at $V_{CC} = 6.0V$ and $V_{pp} = 12.5V$. When the intelligent Programming cycle has been completed, all bytes should be compared to the original data with $V_{CC} = V_{pp} = 5.0V$.

$$\bar{T}_A = 25^\circ\text{C} \pm 5^\circ\text{C}$$

Symbol	Parameter	Limits			Test Conditions
		Min	Max	Unit	
I_{LI}	Input Current (All Inputs)		1.0	μA	$V_{IN} = V_{IL} \text{ or } V_{IH}$
V_{IL}	Input Low Level (All Inputs)	-0.1	0.8	V	
V_{IH}	Input High Level	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage During Verify		0.45	V	$I_{OL} = 2.1 \text{ mA}$
V_{OH}	Output High Voltage During Verify	3.5		V	$I_{OH} = -2.5 \text{ mA}$
$I_{CC2}^{(4)}$	V_{CC} Supply Current		30	mA	
$I_{PP2}^{(4)}$	V_{PP} Supply Current (Program)		50	mA	$\overline{CE} = V_{IL}$
V_{ID}	A_g Intelligent Identifier Voltage	11.5	12.5	V	
V_{PP}	intelligent Programming Algorithm	12.0	13.0	V	
	Quick-Pulse Programming Algorithm	12.5	13.0	V	
V_{CC}	intelligent Programming Algorithm	5.75	6.25	V	
	Quick-Pulse Programming Algorithm	6.0	6.5	V	

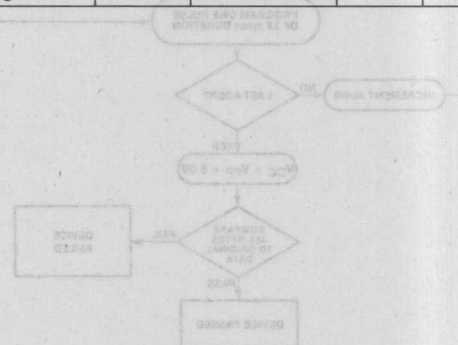


Figure 8. Intelligent Programming™ Flowchart

Intelligent Programming™ Algorithm

The Intelligent Programming Algorithm has been a standard in the industry for the past few years. A flowchart of the Intelligent Programming Algorithm is shown in Figure 8.

The Intelligent Programming Algorithm utilizes two different pulse types: initial and overprogram. The duration of the initial pulse(s) is one millisecond, which will then be followed by a target overprogram pulse of length $3X$ msec. X is an iteration counter

and is equal to the number of the initial one millisecond pulses applied to a particular location. Before a correct verify occurs, up to 25 one-millisecond pulses are provided for before the overprogram pulse is applied.

The entire sequence of program pulses and data verification is performed at $V_{CC} = 6.0\text{V}$ and $V_{PP} = 12.5\text{V}$. When the Intelligent Programming cycle has been completed, all bytes should be compared to the original data with $V_{CC} = V_{PP} = 6.0\text{V}$.

A.C. PROGRAMMING CHARACTERISTICS 27C256

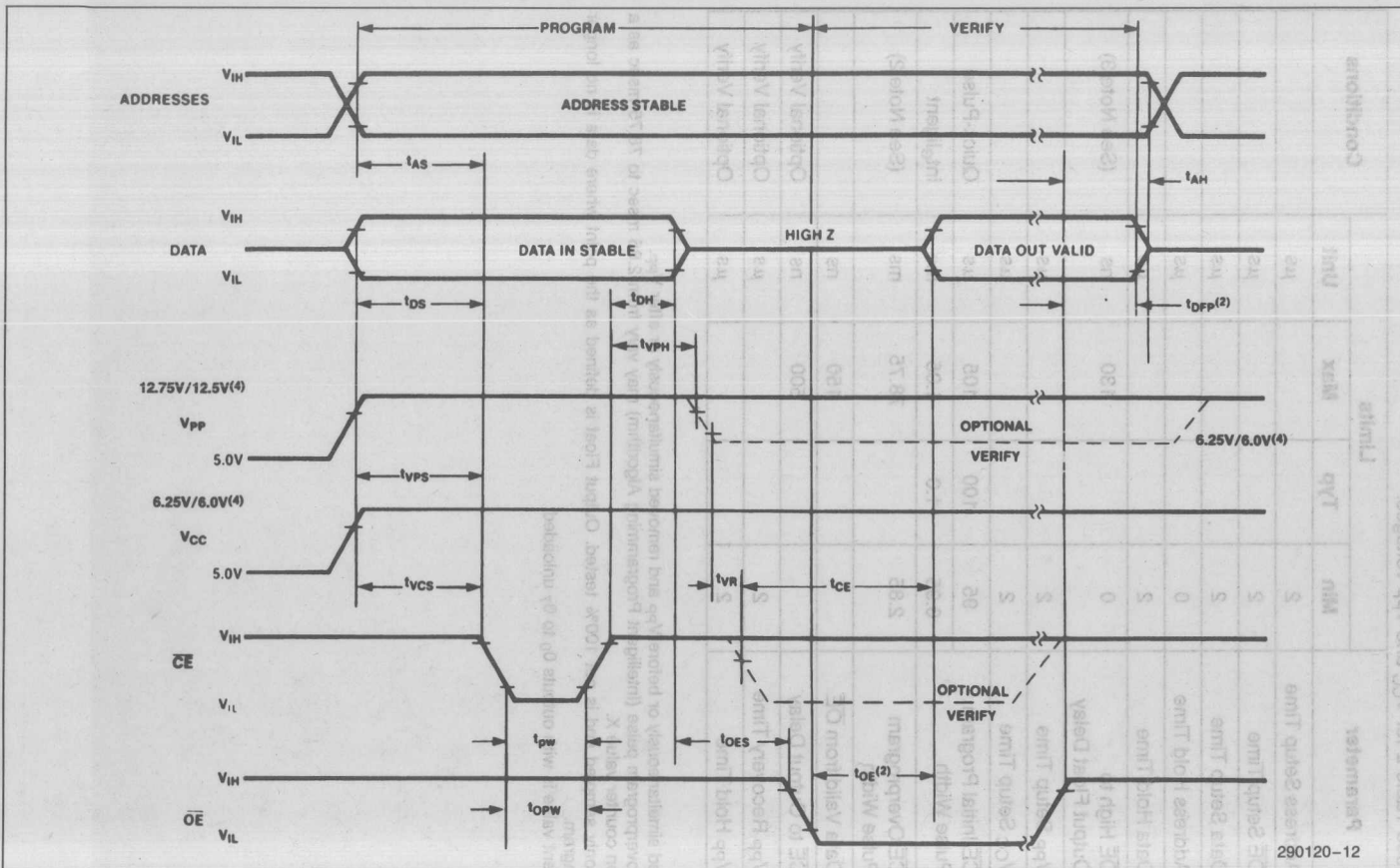
T_A = 25°C ± 5°C; see Table 2 for V_{CC} and V_{pp} voltages.

Symbol	Parameter	Limits				Conditions
		Min	Typ	Max	Unit	
t _{AS}	Address Setup Time	2			μs	
t _{OES}	\overline{OE} Setup Time	2			μs	
t _{DS}	Data Setup Time	2			μs	
t _{AH}	Address Hold Time	0			μs	
t _{DH}	Data Hold Time	2			μs	
t _{DFP}	\overline{OE} High to Output Float Delay	0		130	ns	(See Note 3)
t _{VPS}	V _{pp} Setup Time	2			μs	
t _{VCS}	V _{CC} Setup Time	2			μs	
t _{PW}	\overline{CE} Initial Program Pulse Width	95	100	105	μs	Quick-Pulse
		0.95	1.0	1.05	ms	intelligent
t _{OPW}	\overline{CE} Overprogram Pulse Width	2.85		78.75	ms	(See Note 2)
t _{OE}	Data Valid from \overline{OE}			150	ns	
t _{CE}	\overline{CE} to Output Delay			500	ns	Optional Verify
t _{VR}	V _{pp} Recovery Time	2			μs	Optional Verify
t _{VPH}	V _{pp} Hold Time	2			μs	Optional Verify

NOTES:

1. V_{CC} must be applied simultaneously or before V_{pp} and removed simultaneously or after V_{pp}.
2. The length of the overprogram pulse (Intelligent Programming Algorithm) may vary from 2.85 msec to 78.75 msec as a function of the iteration counter value X.
3. This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.
4. The maximum current value is with outputs O₀ to O₇ unloaded.

PROGRAMMING WAVEFORMS 27C256



NOTES:

1. The Input Timing Reference Level is 0.8V for V_{IL} and 2V for V_{IH} .
2. t_{OE} and t_{DFP} are characteristics of the device but must be accommodated by the programmer.
3. When programming the 27C256, a 0.1 μ F capacitor is required across V_{PP} and ground to suppress spurious voltage transients which can damage the device.
4. 12.75V V_{PP} & 6.25V V_{CC} for Quick-Pulse Programming Algorithm; 12.5V V_{PP} & 6.0V V_{CC} for intelligent Programming Algorithm.

290120-12



725078



87C257 256K (32K x 8) CHMOS UV ERASABLE PROM

Automotive

- **Extended Automotive Temperature Range:** -40°C to $+125^{\circ}\text{C}$
- **CHMOS/NMOS Microcontroller and Microprocessor Compatible**
 - 87C257-Integrated Address Latch
 - Universal 28 Pin Memory Site, 2-line Control
- **Noise Immunity Features**
 - $\pm 10\%$ V_{CC} Tolerance
 - Maximum Latch-up Immunity Through EPI Processing
- **New Quick-Pulse Programming™ Algorithm**
 - 4 Second Programming
- **Low Power Consumption**
- **High Performance Speeds**
 - 200 ns Maximum Access Time
- **Available in 28-Pin Cerdip Package**
(See Packaging Spec., Order #231369)

Intel's 87C257 CHMOS EPROM is a 256K-bit 5V only memory organized as 32,768 8-bit words. It employs advanced CHMOS*II-E circuitry for systems requiring low power, high speed performance, and noise immunity. The 87C257 is optimized for compatibility with multiplexed address/data bus microcontrollers such as Intel's 16 MHz 80C51, 80C152, 80C252, and 8 MHz 80C196.

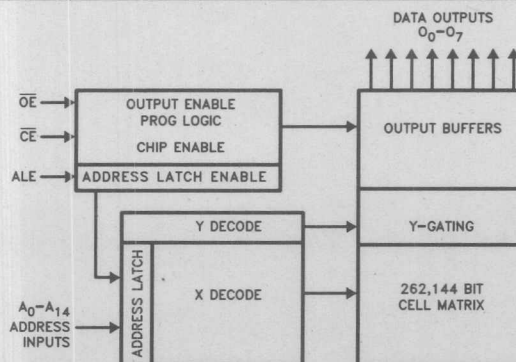
The 87C257 incorporates latches on all address inputs to minimize chip count, reduce cost, and simplify design of multiplexed bus systems. The 87C257's internal address latch allows address and data pins to be tied directly to the processor's multiplexed address/data pins. Address information (inputs A_0-A_{14}) is latched early in the memory-fetch cycle by the falling edge of the ALE input. Subsequent address information is ignored while ALE remains low. The EPROM can then pass data (from pins O_0-O_7) on the same bus during the last part of the memory-fetch cycle.

The 87C257 is offered in a ceramic DIP package, providing flexibility in prototyping and R&D environments. The 87C257 employs the Quick-Pulse Programming™ Algorithm for fast and reliable programming.

Intel's EPI processing achieves the highest degree of latch-up protection. Address and data pin latch-up prevention is provided for stresses up to 100 mA from -1V to $V_{CC} + 1\text{V}$.

In order to meet the rigorous environmental requirements of automotive applications, Intel offers the 87C257 in extended Automotive temperature range. Operational characteristics are guaranteed over the range of -40°C to $+125^{\circ}\text{C}$ ambient.

*HMOS and CHMOS are patented processes of Intel Corporation.



290142-1

Figure 1. Block Diagram

Pin Names

A ₀ -A ₁₄	ADDRESSES
O ₀ -O ₇	OUTPUTS
OE	OUTPUT ENABLE
CE	CHIP ENABLE
ALE/V _{PP}	Address Latch Enable/V _{PP}
N.C.	NO CONNECT

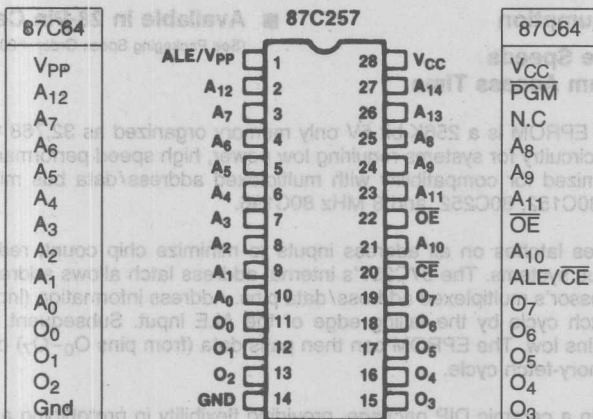


Figure 2. DIP Pin Configuration

NOTE: Intel "Universal Site"-Compatible EPROM Pin Configurations are Shown in the Blocks Adjacent.

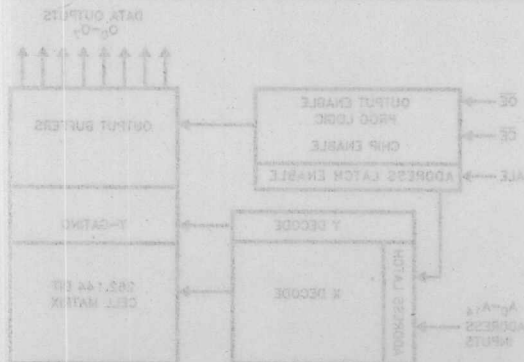


Figure 1. Block Diagram

ABSOLUTE MAXIMUM RATINGS*

Operating Temperature During Read	−40°C to +125°C
Temperature Under Bias	−40°C to +125°C
Storage Temperature	−65°C to +150°C
Voltage on any Pin with Respect to Ground	−2V to +7V(1)
Voltage on A ₉ with Respect to Ground	−2V to +13.5V(1)
V _{PP} Supply Voltage with Respect to Ground During Programming	−2V to +14.0V(1)
V _{CC} Supply Voltage with Respect to Ground	−2V to +7.0V(1)

Maximum Junction Temperature (T_J) 140°C

Maximum Thermal Resistance

Junction to Ambient (θ_{JA}):

Cerdip 40°C/W

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

READ OPERATION

D.C. CHARACTERISTICS TTL and NMOS Inputs −40°C ≤ T_A ≤ +125°C

Symbol	Parameter	Notes	Min	Typ(2)	Max	Units	Test Condition
I _{LI}	Input Load Current			0.01	±1.0	μA	V _{IN} = 0V, 5.5V
I _{LO}	Output Leakage Current			0.01	±10	μA	V _{OUT} = 0V, 5.5V
I _{SB}	V _{CC} Current Standby with Inputs—	Switching			10	mA	CE = ALE = V _{IH}
		Stable			1.0	mA	CE = V _{IH} , ALE = V _{IL}
I _{CC1}	V _{CC} Current Active	4			30	mA	CE = V _{IL} , ALE = V _{IH} f = 5 MHz, I _{OUT} = 0 mA
V _{IL}	Input Low Voltage (±10% Supply)	1	−0.5		0.8	V	
V _{IH}	Input High Voltage (±10% Supply)		2.0		V _{CC} + 0.5	V	
V _{OL}	Output Low Voltage				0.45	V	I _{OL} = 2.1 mA
V _{OH}	Output High Voltage		2.4			V	I _{OH} = −400 μA
I _{OS}	Output Short Circuit Current	5			100	mA	

D.C. CHARACTERISTICS CMOS Inputs −40°C ≤ T_A ≤ +125°C

Symbol	Parameter	Notes	Min	Typ(2)	Max	Units	Test Condition
I _{LI}	Input Load Current			0.01	±1.0	μA	V _{IN} = 5.5V
I _{LO}	Output Leakage Current			0.01	±10.0	μA	V _{OUT} = 0V, 5.5V
I _{SB}	V _{CC} Current Standby with Inputs—	Switching			6	mA	CE = ALE = V _{IH}
		Stable			100	μA	CE = V _{IH} , ALE = V _{IL}
I _{CC1}	V _{CC} Current Active	4			15	mA	CE = V _{IL} , ALE = V _{IH} f = 5 MHz, I _{OUT} = 0 mA
V _{IL}	Input Low Voltage (±10% Supply)		−0.2		0.8	V	
V _{IH}	Input High Voltage (±10% Supply)		0.7 V _{CC}		V _{CC} + 0.2	V	
V _{OL}	Output Low Voltage				0.4	V	I _{OL} = 2.1 mA
V _{OH}	Output High Voltage		V _{CC} − 0.8			V	I _{OH} = −2.5 mA
I _{OS}	Output Short Circuit Current	5			100	mA	

NOTES:

1. Minimum D.C. input voltage is −0.5V. During transitions, the inputs may undershoot to −2.0V for periods less than 20 ns. Maximum D.C. voltage on output pins is V_{CC} + 0.5V which may overshoot to V_{CC} + 2V for periods less than 20 ns.
2. Typical limits are at V_{CC} = 5V, T_A = +25°C.
3. CE is V_{CC} ± 0.2V. All other inputs can have any value within spec.
4. Maximum current value is with outputs O₀ to O₇ unloaded.
5. Output shorted for no more than one second. No more than one output shorted at a time. I_{OS} is sampled but not 100% tested.

READ OPERATION

A.C. CHARACTERISTICS⁽¹⁾ $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$

Versions(3)		V _{CC} ± 5%	87C257-200V05		87C257-250V05		Units
		V _{CC} ± 10%			87C257-250V10		
Symbol	Characteristic	Min	Max	Min	Max		
t _{ACC}	Address to Output Delay		200		250	ns	
t _{CE}	\overline{CE} to Output Delay		200		250	ns	
t _{OE}	\overline{OE} to Output Delay		75		100	ns	
t _{DF} (2)	\overline{OE} High to Output High Z		40		55	ns	
t _{OH} (2)	Output Hold from Addresses, \overline{CE} or \overline{OE} Change-Whichever is First	0		0		ns	
t _{LL}	Latch Deselect Width	55		60		ns	
t _{AL} (2)	Address to Latch Set-Up	15		25		ns	
t _{LA}	Address Hold from LATCH	30		40		ns	
t _{LOE}	ALE to Output Enable	30		40		ns	

NOTES:

1. See A.C. Testing Input/Output Waveforms for timing measurements.

2. Guaranteed and sampled.

3. Model Number Prefixes: No Prefix = Cerdip.

A.C. CONDITIONS OF TEST

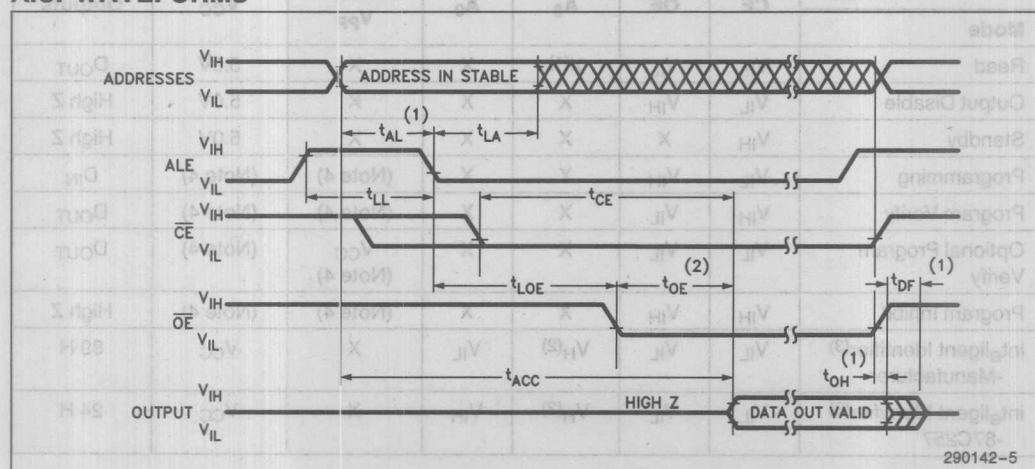
Input Rise and Fall Times (10% to 90%) 10 ns

Input Pulse Levels V_{OL} to V_{OH}

Input Timing Reference Level 1.5V

Output Timing Reference Level V_{IL} and V_{IH}

A.C. WAVEFORMS



NOTES:

1. This parameter is only sampled and is not 100% tested.

2. \overline{OE} may be delayed up to $t_{CE} - t_{OE}$ after the falling edge of \overline{CE} without impact on t_{CE} .

CAPACITANCE(1) $T_A = 25^\circ\text{C}$, $f = 1.0\text{ MHz}$

Symbol	Parameter	Max	Units	Conditions
C_{IN}	Address/Control Capacitance	6	pF	$V_{IN} = 0\text{V}$
C_{OUT}	Output Capacitance	12	pF	$V_{OUT} = 0\text{V}$

NOTE:

1. Sampled. Not 100% tested.

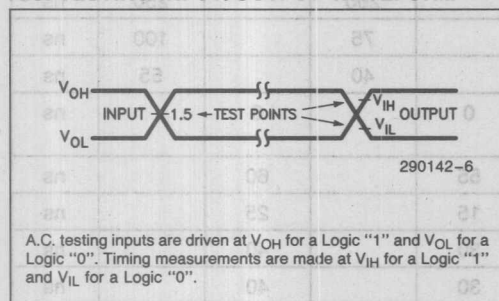
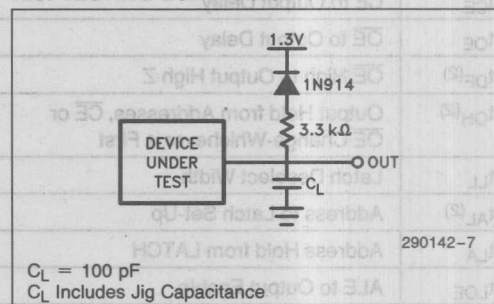
A.C. TESTING INPUT/OUTPUT WAVEFORM**A.C. TESTING LOAD CIRCUIT****DEVICE OPERATION**

Table 1 lists 87C257 operating modes. Read mode requires a single 5V power supply. All input levels are TTL or CMOS except A9 in intelligent Identifier mode and V_{PP} .

Table 1. Mode Selection

Pins	\overline{CE}	\overline{OE}	A_9	A_0	ALE/V_{PP}	V_{CC}	Outputs
Read	V_{IL}	V_{IL}	X(1)	X	X	5.0V	D_{OUT}
Output Disable	V_{IL}	V_{IH}	X	X	X	5.0V	High Z
Standby	V_{IH}	X	X	X	X	5.0V	High Z
Programming	V_{IL}	V_{IH}	X	X	(Note 4)	(Note 4)	D_{IN}
Program Verify	V_{IH}	V_{IL}	X	X	(Note 4)	(Note 4)	D_{OUT}
Optional Program Verify	V_{IL}	V_{IL}	X	X	V_{CC} (Note 4)	(Note 4)	D_{OUT}
Program Inhibit	V_{IH}	V_{IH}	X	X	(Note 4)	(Note 4)	High Z
intelligent Identifier ⁽³⁾ -Manufacturer	V_{IL}	V_{IL}	V_H (2)	V_{IL}	X	V_{CC}	89 H
intelligent Identifier ⁽³⁾ -87C257	V_{IL}	V_{IL}	V_H (2)	V_{IH}	X	V_{CC}	24 H

NOTES:1. X can be V_{IL} or V_{IH} .2. $V_H = 12.0\text{V} \pm 0.5\text{V}$.3. A_1-A_8 , $A_{10}-12 = V_{IL}$, $A_{13}-14 = X$.4. See Table 2 for V_{CC} and V_{PP} programming voltages.

Read Mode

The 87C257 has two control functions; both must be logically active to obtain data at the outputs. Chip Enable (\overline{CE}) is the power control and the device-select. Output enable (\overline{OE}) gates data to the output pins by controlling the output buffer. When the address is stable ($ALE = V_{IH}$) or latched ($ALE = V_{IL}$), the address access time (t_{ACC}) equals the delay from \overline{CE} to output (t_{CE}). Outputs display valid data t_{OE} after the falling edge of \overline{OE} , assuming t_{ACC} and t_{CE} times are met.

The 87C257 reduces the hardware interface in multiplexed address-data bus systems. Figure 4 shows a low power, small board space, minimal chip 87C257/microcontroller design. The processor's multiplexed bus (AD_{0-7}) is tied to the 87C257's address and data pins. No separate address latch is needed because the 87C257 latches all address inputs when ALE is low.

The ALE input controls the 87C257's internal address latch. As ALE transitions from V_{IH} to V_{IL} , the last address present at the address pins is retained. The \overline{OE} control can then enable EPROM data onto the bus.

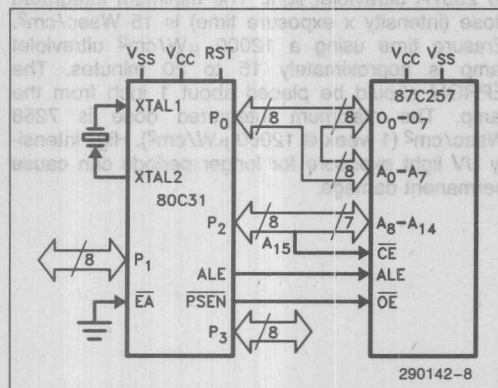


Figure 4. 80C31 with 87C257 System Configuration

Standby Mode

The standby mode substantially reduces V_{CC} current. When $\overline{CE} = V_{IH}$, the standby mode places the outputs in a high impedance state, independent of the \overline{OE} input.

Two Line Output Control

EPROMs are often used in larger memory arrays. Intel provides two control inputs to accommodate multiple memory connections. Two-line control provides for:

- the lowest possible memory power dissipation, and
- complete assurance that output bus contention will not occur.

To efficiently use these two control inputs, an address decoder should enable \overline{CE} , while \overline{OE} should be connected to all memory-array devices and the system's READ control line. This assures that only selected memory devices have active outputs while deselected memory devices are in low-power standby mode.

SYSTEM CONSIDERATIONS

EPROM power switching characteristics require careful device decoupling. System designers are interested in three supply current (ICC) issues—standby current levels, active current levels, and transient current peaks produced by falling and rising edges of Chip Enable. Transient current magnitudes depend on the device outputs' capacitive and inductive loading. Two-Line Control and proper decoupling capacitor selection will suppress transient voltage peaks. Each device should have a 0.1 μF ceramic capacitor connected between its V_{CC} and GND. This high frequency, low inherent-inductance capacitor should be placed as close as possible to the device. Additionally, for every eight devices, a 4.7 μF electrolytic capacitor should be placed between V_{CC} and GND at the array's power supply connection. The bulk capacitor will overcome voltage slumps caused by PC board trace inductances.

PROGRAMMING MODES

Caution: Exceeding 14V on V_{pp} will permanently damage the device.

Initially, and after each erasure, all EPROM bits are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" are programmed, the data word

can contain both "1s" and "0s". Ultraviolet light erasure is the only way to change "0s" to "1s".

The programming mode is entered when V_{PP} is raised to its programming voltage (see Table 2). Data is programmed by applying an 8-bit word to the output pins (O_0 – O_7). Pulsing \overline{CE} to TTL-low while $\overline{OE} = V_{IH}$ will program data. TTL levels are required for address and data inputs.

Program Inhibit

The Program Inhibit mode allows parallel programming of multiple EPROMs with different data. With V_{PP} at its programming voltage, a \overline{CE} -low pulse programs the desired EPROM. \overline{CE} -high inputs inhibit programming of non-targeted devices. Except for \overline{CE} and \overline{OE} , parallel EPROMs may have common inputs.

Program Verify

With V_{PP} and V_{CC} at their programming voltages, a verify (read) determines that bits are correctly programmed. The verify is performed with $\overline{CE} = V_{IH}$ and $\overline{OE} = V_{IL}$. Valid data is available t_{OE} after \overline{OE} falls low.

Intelligent Identifier™ Mode

The Intelligent Identifier Mode will determine an EPROM's manufacturer and device type. Program-

ming equipment can automatically match a device with its proper programming algorithm.

This mode is activated when programming equipment forces $12V \pm 0.5V$ on the EPROM's A_9 address line. With A_1 – A_8 , A_{10} – $A_{12} = V_{IL}$ (A_{13} – A_{14} are don't care), address line $A_0 = V_{IL}$ will present the manufacturer's code and $A_0 = V_{IH}$ the device code (see Table 1). When $A_9 = V_{IH}$, ALE need not be toggled to latch each identifier address. This mode functions in the $25^\circ\text{C} \pm 5^\circ\text{C}$ ambient temperature range required during programming.

ERASURE CHARACTERISTICS (FOR Cerdip EPROMs)

Exposure to light of wavelength shorter than 4000 Angstroms (\AA) begins EPROM erasure. Sunlight and some fluorescent lamps have wavelengths in the 3000–4000 \AA range. Constant exposure to room-level fluorescent light can erase an EPROM in about 3 years (about 1 week for direct sunlight). Opaque labels over the window will prevent unintentional erasure under these lighting conditions.

The recommended erasure procedure is exposure to 2537 \AA ultraviolet light. The minimum integrated dose (intensity x exposure time) is 15 Wsec/cm². Erasure time using a 12000 $\mu\text{W}/\text{cm}^2$ ultraviolet lamp is approximately 15 to 20 minutes. The EPROM should be placed about 1 inch from the lamp. The maximum integrated dose is 7258 Wsec/cm² (1 week @ 12000 $\mu\text{W}/\text{cm}^2$). High intensity UV light exposure for longer periods can cause permanent damage.

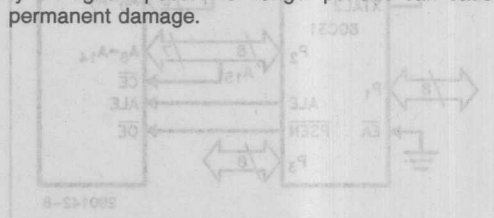


Figure 4. 87C257 with 87C257 System Configuration

Standby Mode

The standby mode substantially reduces V_{CC} current. When $\overline{OE} = V_{IH}$, the standby mode places the outputs in a high impedance state, independent of the \overline{OE} input.

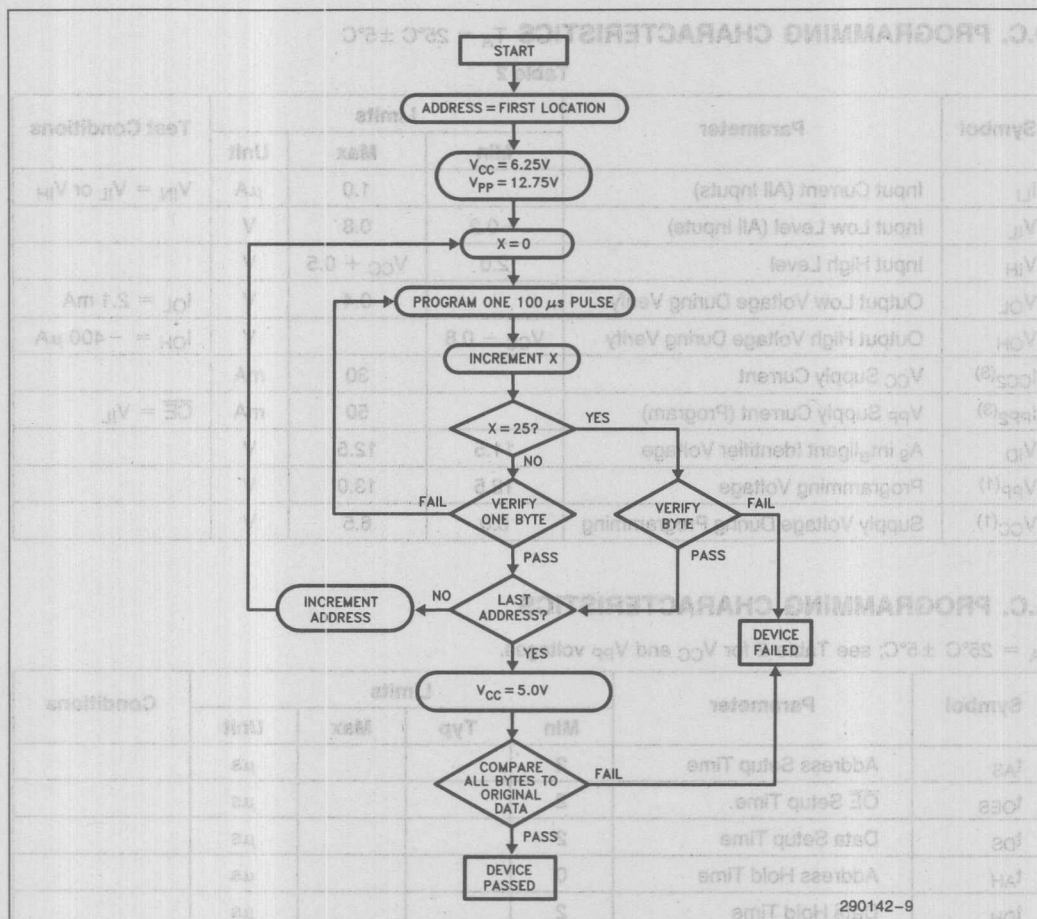


Figure 5. Quick-Pulse Programming™ Algorithm

CHMOS NOISE CHARACTERISTICS

System reliability is enhanced by Intel's CHMOS EPI-process techniques. Protection on each data and address pin prevents latch-up; even with 100 mA currents and voltages from $-1V$ to $V_{CC} + 1V$. Additionally, the V_{PP} pin is designed to resist latch-up to the 14V maximum device limit.

Quick-Pulse Programming™ Algorithm

The Quick-Pulse Programming algorithm programs Intel's 87C257 EPROM. Developed to substantially reduce production programming throughput time, this algorithm can program a 87C257 in under four seconds. Actual programming time depends on the PROM programmer used.

The Quick-Pulse Programming algorithm uses a 100 microsecond initial-pulse followed by a byte verifica-

tion to determine when the addressed byte is correctly programmed. The algorithm terminates if 25 $100\mu s$ pulses fail to program a byte. Figure 5 shows the Quick-Pulse Programming algorithm flowchart.

The entire program-pulse/byte-verify sequence is performed with $V_{CC} = 6.25V$ and $V_{PP} = 12.75V$. When programming is complete, all bytes should be compared to the original data with $V_{CC} = 5.0V$.

Alternate Programming

Intel's 27C256 and 27256 Quick-Pulse Programming algorithms will also program the 87C257. By overriding a check for the intelligent Identifier, older or non-upgraded PROM programmers can program the 87C257. See Intel's 27C256 and 27256 data sheets for programming waveforms of these alternate algorithms.

D.C. PROGRAMMING CHARACTERISTICS $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$

Table 2

Symbol	Parameter	Limits			Test Conditions
		Min	Max	Unit	
I_{LI}	Input Current (All Inputs)		1.0	μA	$V_{IN} = V_{IL} \text{ or } V_{IH}$
V_{IL}	Input Low Level (All Inputs)	-0.2	0.8	V	
V_{IH}	Input High Level	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage During Verify		0.4	V	$I_{OL} = 2.1 \text{ mA}$
V_{OH}	Output High Voltage During Verify	$V_{CC} - 0.8$		V	$I_{OH} = -400 \mu\text{A}$
$I_{CC2}^{(3)}$	V_{CC} Supply Current		30	mA	
$I_{PP2}^{(3)}$	V_{PP} Supply Current (Program)		50	mA	$\overline{CE} = V_{IL}$
V_{ID}	A_9 intelligent Identifier Voltage	11.5	12.5	V	
$V_{PP}^{(1)}$	Programming Voltage	12.5	13.0	V	
$V_{CC}^{(1)}$	Supply Voltage During Programming	6.0	6.5	V	

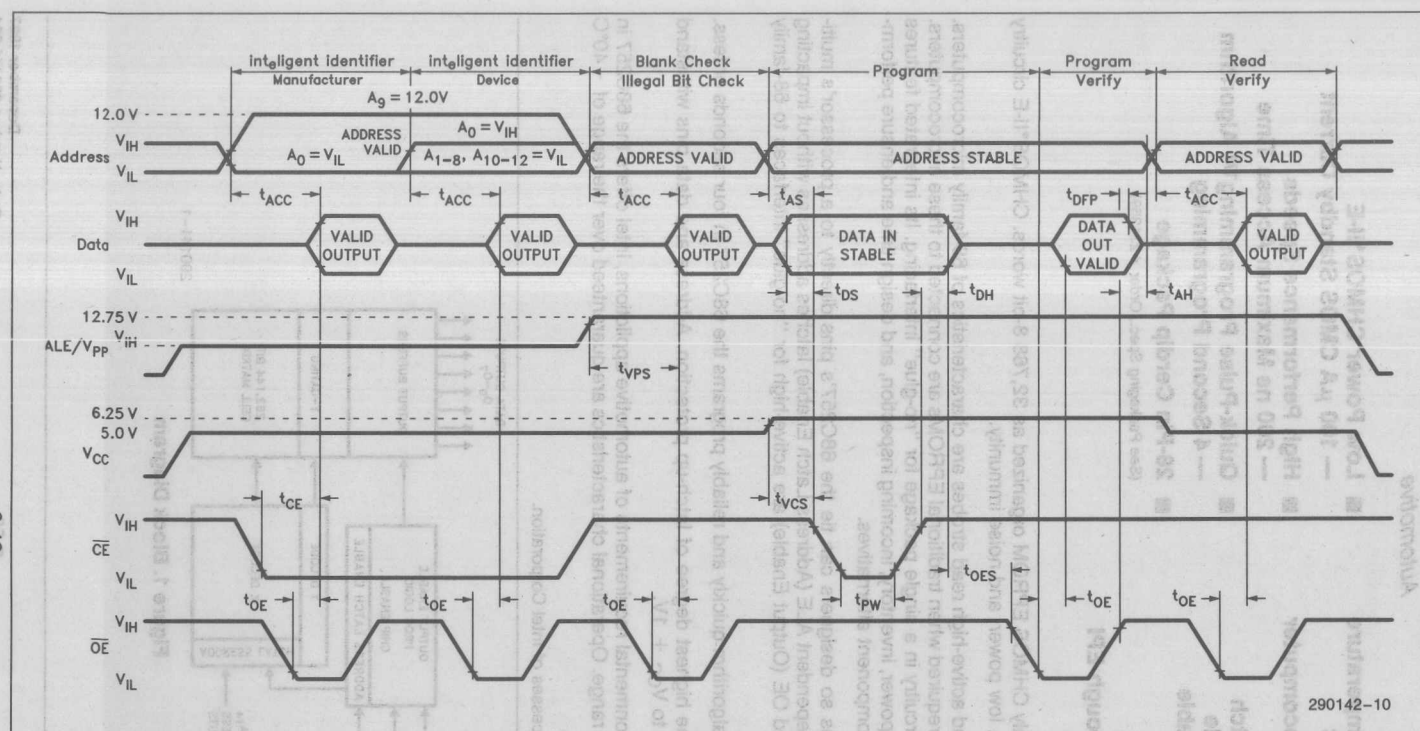
A.C. PROGRAMMING CHARACTERISTICS $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$; see Table 2 for V_{CC} and V_{PP} voltages.

Symbol	Parameter	Limits				Conditions
		Min	Typ	Max	Unit	
t_{AS}	Address Setup Time	2			μs	
t_{OES}	\overline{OE} Setup Time	2			μs	
t_{DS}	Data Setup Time	2			μs	
t_{AH}	Address Hold Time	0			μs	
t_{DH}	Data Hold Time	2			μs	
$t_{DFP}^{(2)}$	\overline{OE} High to Output Float Delay	0		130	ns	
$t_{VPS}^{(1)}$	V_{PP} Setup Time	2			μs	
$t_{VCS}^{(1)}$	V_{CC} Setup Time	2			μs	
t_{PW}	\overline{CE} Program Pulse Width	95	100	105	μs	
t_{OE}	Data Valid from \overline{OE}			150	ns	

NOTES:

1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP} .
2. This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.
3. The maximum current value is with outputs O_0 to O_7 unloaded.

PROGRAMMING WAVEFORMS



NOTES:

1. The input timing reference level is $V_{IL} = 0.8V$ and $V_{IH} = 2V$.
2. t_{OE} and t_{DFP} are device characteristics but must be accommodated by the programmer.
3. To prevent device damage during programming, a $0.1 \mu F$ capacitor is required between V_{PP} and ground to suppress spurious voltage transients.
4. During programming, the address latch function is bypassed whenever $V_{PP} = 12.75V$ or $A_9 = V_{IH}$. When V_{PP} and A_9 are at TTL levels, the address latch function is enabled, and the device functions in read mode.
5. V_{PP} can be 12.75V during Blank Check and Final Verify; if so, \overline{CE} must be V_{IH} .

68C257 256K (32K x 8) CHMOS UV ERASABLE PROM

Automotive

- **Extended Automotive Temperature Range:** -40°C to $+125^{\circ}\text{C}$
- **6801, 6803, 68HC11 Microcomputer Compatible**
 - Integrated Address Latch
 - Active-High Chip Enable
 - Active-High Output Enable
- **Noise Immunity Features**
 - $\pm 10\%$ V_{CC} Tolerance
 - Latch-up Immunity Through EPI Processing
- **Low Power CHMOS*II-E**
 - 100 μA CMOS Standby Current
- **High Performance Speeds**
 - 200 ns Maximum Access Time
- **Quick-Pulse Programming™ Algorithm**
 - 4 Second Programming
- **28-Pin Cerdip Package**
 - (See Packaging Spec., Order #231369)

Intel's 68C257 is a 256K-bit 5V only CHMOS EPROM organized as 32,768 8-bit words. CHMOS*II-E circuitry provides high speed performance, low power, and noise immunity.

High-memory boot-up locations and active-high read strobes are characteristics of 68-family microcomputers. Address latches and inverters are required when traditional EPROMs are connected to these microcomputers. The 68C257 contains all of this circuitry in a single package for "no-glue" interfacing. Its integrated features reduce system cost, board space, power, inventory, incoming inspection, and design time and ensure performance that is superior to discrete-component alternatives.

Address inputs incorporate latches so designers can tie the 68C257's pins directly to a processor's multiplexed address/data pins. The independent ALE (Address Latch Enable) latches addresses without impacting access time. CE (Chip Enable) and OE (Output Enable) are active-high for "no-glue" interfaces to 68-family microcomputers.

The Quick-Pulse Programming™ algorithm quickly and reliably programs the 68C257 in four seconds or less.

Intel's EPI processing achieves the highest degree of latch-up protection. Address and data pins withstand stresses up to 100 mA from -1V to $V_{CC} + 1\text{V}$.

In order to meet the rigorous environmental requirements of automotive applications, Intel offers the 68C257 in extended Automotive temperature range. Operational characteristics are guaranteed over the range of -40°C to $+125^{\circ}\text{C}$ ambient.

*HMOS and CHMOS are patented processes of Intel Corporation.

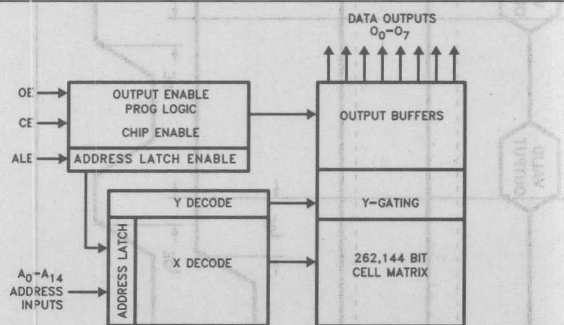
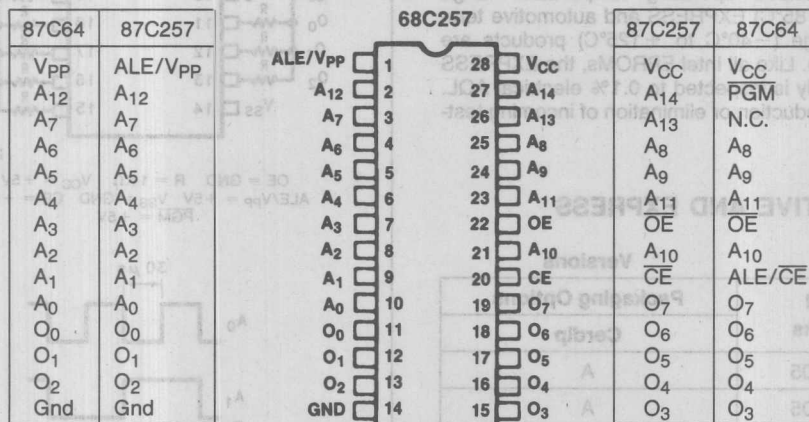


Figure 1. Block Diagram

Pin Names

A ₀ -A ₁₄	ADDRESSES
O ₀ -O ₇	OUTPUTS
OE, \overline{OE}	OUTPUT ENABLE
CE, \overline{CE}	CHIP ENABLE
ALE/V _{PP}	ADDRESS LATCH ENABLE/V _{PP}
N.C.	NO CONNECT
D.U.	DON'T USE



290141-2

Figure 2. DIP Pin Configuration

NOTE:

Intel "Universal Site"-Compatible EPROM Pin Configurations are Shown in the Blocks Adjacent.

Type	Operating Temperature (°C)	Burn-in 125°C (hr)
Q	0°C to +70°C	168 ± 8
T	-40°C to +85°C	NONE
L	-40°C to +85°C	168 ± 8
A	-40°C to +125°C	NONE
B	-40°C to +125°C	168 ± 8

(EXPRESS) EPROMs

The Intel EXPRESS EPROM family receives additional processing to enhance product characteristics. EXPRESS processing is available for several EPROM densities allowing the appropriate memory size to match system applications. EXPRESS EPROMs are available with 168 \pm 8 hour, 125°C dynamic burn-in using Intel's standard bias configuration. This process meets or exceeds most industry burn-in specifications. The standard EXPRESS EPROM operating temperature range is 0°C to +70°C. Extended operating temperature range (-40°C to +85°C) EXPRESS and automotive temperature range (-40°C to +125°C) products are also available. Like all Intel EPROMs, the EXPRESS EPROM family is inspected to 0.1% electrical AQL. This allows reduction or elimination of incoming testing.

AUTOMOTIVE AND EXPRESS OPTIONS

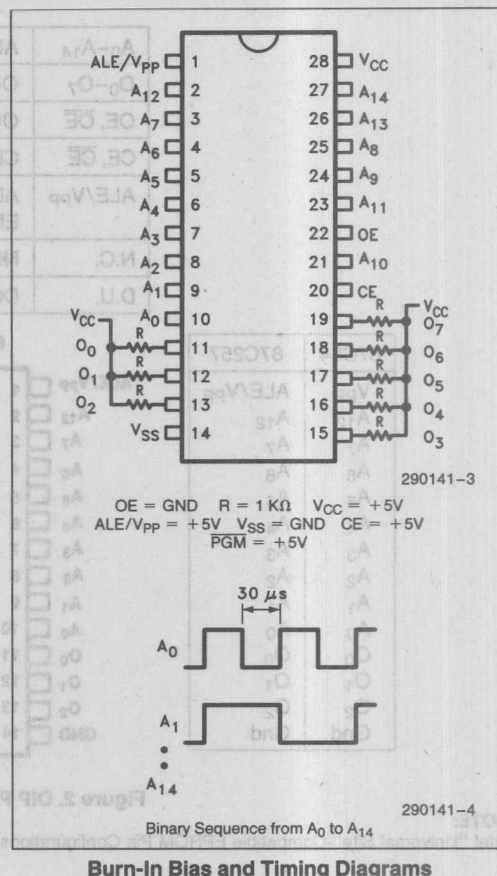
Versions

Speed Versions	Packaging Options	
	Cerdip	
-200V05	A	
-250V05	A	
-250V10	A	

AUTOMOTIVE AND EXPRESS EPROM PRODUCT FAMILY

PRODUCT DEFINITIONS

Type	Operating Temperature (°C)	Burn-in 125°C (hr)
Q	0°C to +70°C	168 \pm 8
T	-40°C to +85°C	NONE
L	-40°C to +85°C	168 \pm 8
A	-40°C to +125°C	NONE
B	-40°C to +125°C	168 \pm 8



ABSOLUTE MAXIMUM RATINGS*

Operating Temperature	
During Read	-40°C to +125°C
Temperature Under Bias	-40°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin with	
Respect to Ground	-2V to +7V(1)
Voltage on A ₉ with	
Respect to Ground	-2V to +13.5V(1)
V _{PP} Supply Voltage with	
Respect to Ground	
During Programming	-2V to +14.0V(1)
V _{CC} Supply Voltage with	
Respect to Ground	-2V to +7.0V(1)

Maximum Junction Temperature (T _J)	140°C
Maximum Thermal Resistance	
Junction-to-Ambient (θ _{JA})	
Cerdip	40°C/W

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

READ OPERATION

D.C. CHARACTERISTICS TTL and NMOS Inputs -40°C ≤ T_A ≤ +125°C

Symbol	Parameter	Notes	Min	Typ(2)	Max	Units	Test Condition
I _{LI}	Input Load Current			0.01	±1.0	μA	V _{IN} = 0V, 5.5V
I _{LO}	Output Leakage Current			0.01	±10	μA	V _{OUT} = 0V, 5.5V
I _{SB}	V _{CC} Current Standby with Inputs—	Switching			10	mA	CE = V _{IL} , ALE = V _{IH}
		Stable			1.0	mA	CE = ALE = V _{IL}
I _{CC1}	V _{CC} Current Active	4			30	mA	CE = ALE = V _{IH} f = 5 MHz, I _{OUT} = 0 mA
V _{IL}	Input Low Voltage (±10% Supply)	1	-0.5		0.8	V	
V _{IH}	Input High Voltage (±10% Supply)		2.0		V _{CC} + 0.5	V	
V _{OL}	Output Low Voltage				0.45	V	I _{OL} = 2.1 mA
V _{OH}	Output High Voltage		2.4			V	I _{OH} = -400 μA
I _{OS}	Output Short Circuit Current	5			100	mA	

D.C. CHARACTERISTICS CMOS Inputs -40°C ≤ T_A ≤ +125°C

Symbol	Parameter	Notes	Min	Typ(2)	Max	Units	Test Condition
I _{LI}	Input Load Current			0.01	±1.0	μA	V _{IN} = 5.5V
I _{LO}	Output Leakage Current			0.01	±10.0	μA	V _{OUT} = 0V, 5.5V
I _{SB}	V _{CC} Current Standby with Inputs—	Switching	3		6	mA	CE = V _{IL} , ALE = V _{IH}
		Stable			100	μA	CE = ALE = V _{IL}
I _{CC1}	V _{CC} Current Active	4			15	mA	CE = ALE = V _{IH} f = 5 MHz, I _{OUT} = 0 mA
V _{IL}	Input Low Voltage (±10% Supply)		-0.2		0.8	V	
V _{IH}	Input High Voltage (±10% Supply)		0.7 V _{CC}		V _{CC} + 0.2	V	
V _{OL}	Output Low Voltage				0.4	V	I _{OL} = 2.1 mA
V _{OH}	Output High Voltage		V _{CC} - 0.8			V	I _{OH} = -2.5 mA
I _{OS}	Output Short Circuit Current	5			100	mA	

NOTES:

1. Minimum D.C. input voltage is -0.5V. During transitions, the inputs may undershoot to -2.0V for periods less than 20 ns. Maximum D.C. voltage on output pins is V_{CC} + 0.5V which may overshoot to V_{CC} + 2V for periods less than 20 ns.
2. Typical limits are at V_{CC} = 5V, T_A = +25°C.
3. CE is ±0.2V. All other inputs can have any value within spec.
4. Maximum current value is with outputs O₀ to O₇ unloaded.
5. Output shorted for no more than one second. No more than one output shorted at a time. I_{OS} is sampled but not 100% tested.

READ OPERATION

A.C. CHARACTERISTICS(1) $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$

Symbol	Characteristic	Versions(3)		68C257-200V05		68C257-250V05		Unit
		$V_{CC} \pm 5\%$ $V_{CC} \pm 10\%$		Min	Max	Min	Max	
t_{ACC}	Address to Output Delay				200		250	ns
t_{CE}	CE to Output Delay				200		250	ns
t_{OE}	OE to Output Delay				75		100	ns
$t_{DF}^{(2)}$	OE Low to Output High Z				40		55	ns
$t_{OH}^{(2)}$	Output Hold from Addresses, CE or OE Change-Whichever is First			0		0		ns
t_{LL}	Latch Deselect Width			55		60		ns
$t_{AL}^{(2)}$	Address to Latch Set-Up			15		25		ns
t_{LA}	Address Hold from LATCH			30		40		ns
t_{LOE}	ALE to Output Enable			30		40		ns

NOTES:

- See A.C. Testing Input/Output Waveforms for timing measurements.
- Guaranteed and sampled.
- Model Number Prefixes: No Prefix = Cerdip.

A.C. CONDITIONS OF TEST

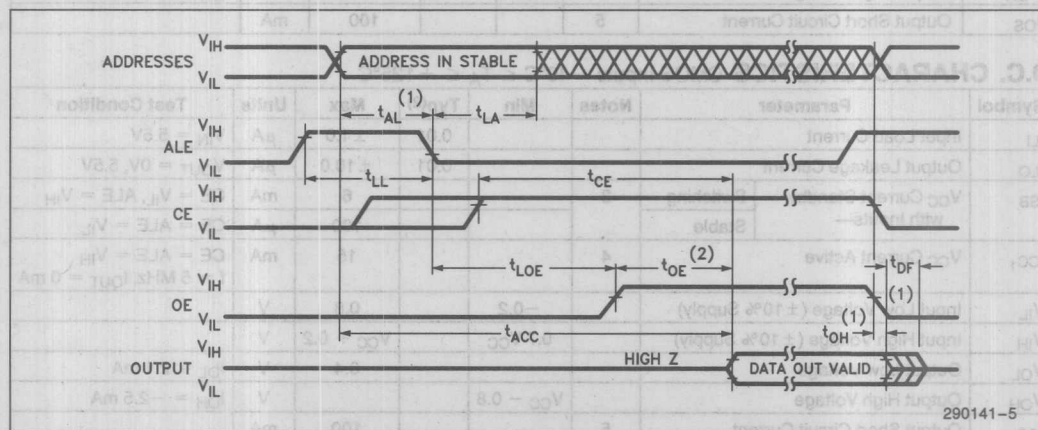
Input Rise and Fall Times (10% to 90%) 10 ns

Input Pulse Levels V_{OL} to V_{OH}

Input Timing Reference Level 1.5V

Output Timing Reference Level V_{IL} and V_{IH}

A.C. WAVEFORMS



NOTES:

- This parameter is only sampled and is not 100% tested.
- OE may be delayed up to $t_{CE} - t_{OE}$ after the rising edge of CE without impact on t_{CE} .

CAPACITANCE(1) $T_A = 25^\circ\text{C}$, $f = 1.0\text{ MHz}$

Symbol	Parameter	Max	Units	Conditions
C_{IN}	Address/Control Capacitance	6	pF	$V_{IN} = 0V$
C_{OUT}	Output Capacitance	12	pF	$V_{OUT} = 0V$

NOTE:

1. Sampled. Not 100% tested.

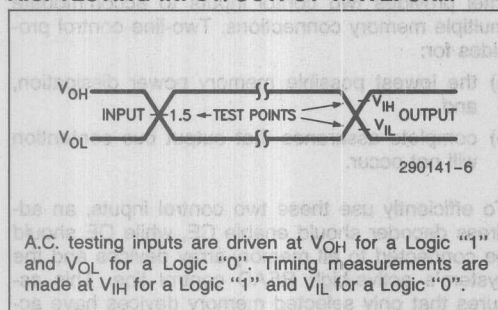
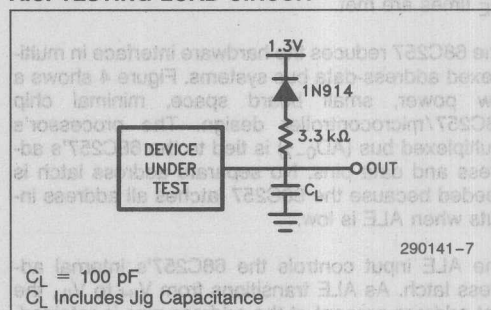
A.C. TESTING INPUT/OUTPUT WAVEFORM**A.C. TESTING LOAD CIRCUIT****DEVICE OPERATION**

Table 1 lists 68C257 operating modes. Read mode requires a single 5V power supply. All input levels are TTL or CMOS except A₉ in intelligent Identifier mode and V_{PP}.

Table 1. Mode Selection

Pins	CE	OE	A ₉	A ₀	ALE/ V _{PP}	V _{CC}	Outputs
Mode							
Read	V_{IH}	V_{IH}	X(1)	X	X	5.0V	D _{OUT}
Output Disable	V_{IH}	V_{IL}	X	X	X	5.0V	High Z
Standby	V_{IL}	X	X	X	X	5.0V	High Z
Programming(5)	V_{IL}	V_{IH}	X	X	(Note 4)	(Note 4)	D _{IN}
Program Verify(5)	V_{IH}	V_{IL}	X	X	(Note 4)	(Note 4)	D _{OUT}
Optional Program Verify	V_{IH}	V_{IH}	X	X	V _{CC} (Note 4)	(Note 4)	D _{OUT}
Program Inhibit(5)	V_{IH}	V_{IH}	X	X	(Note 4)	(Note 4)	High Z
intelligent Identifier(3, 5) -Manufacturer	V_{IL}	V_{IL}	V_H (2)	V_{IL}	X	V _{CC}	89 H
intelligent Identifier(3, 5) -68C257	V_{IL}	V_{IL}	V_H (2)	V_{IH}	X	V _{CC}	27 H

NOTES:1. X can be V_{IL} or V_{IH} .2. $V_H = 12.0V \pm 0.5V$.3. A₁-A₈, A₁₀-A₁₂ = V_{IL} , A₁₃-A₁₄ = X.4. See Table 2 for V_{CC} and V_{PP} programming voltages.5. During Intelligent Identifier Mode (A₉ = V_H) and Programming Modes (ALE/V_{PP} = 12.75V), CE and OE default to active-low enables.

PROGRAMMING MODES

Caution: Exceeding 14V on V_{PP} will permanently damage the device.

Initially, and after each erasure, all EPROM bits are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" are programmed, the data word can contain both "1s" and "0s". Ultraviolet light erasure is the only way to change "0s" to "1s".

The programming mode is entered when V_{PP} is raised to its programming voltage (see Table 2). Data is programmed by applying an 8-bit word to the output pins (O_0 – O_7). Pulsing CE to TTL-low while $OE = V_{IH}$ will program data. TTL levels are required for address and data inputs.

To simplify programming, CE and OE inputs default to active-low (as opposed to read modes active-high) enables when either $A_9 = V_{IH}$ or ALE/V_{PP} is at its programming voltage (12.75V). This allows programming equipment to identify the 68C257 and program it with a standard programming algorithm. The 68C257 uses the same programming algorithm as the 87C257. Programming equipment that programs one device can also program its counterpart.

Program Inhibit

The Program Inhibit mode allows parallel programming of multiple EPROMs with different data. With V_{PP} at its programming voltage, a CE-low pulse programs the desired EPROM. CE-high inputs inhibit programming of non-targeted devices. Except for CE and OE, parallel EPROMs may have common inputs.

Program Verify

With V_{PP} and V_{CC} at their programming voltages, a verify (read) determines that bits are correctly programmed. The verify is performed with $CE = V_{IH}$ and $OE = V_{IL}$. Valid data is available t_{OE} after OE falls low.

intelligent Identifier™ Mode

The intelligent Identifier Mode will determine an EPROM's manufacturer and device type. Programming equipment can automatically match a device with its proper programming algorithm.

This mode is activated when programming equipment forces $12V \pm 0.5V$ on the EPROM's A_9 address line. With A_1 – A_8 , A_{10} – $A_{12} = V_{IL}$ (A_{13} – A_{14} are don't-care), address line $A_0 = V_{IL}$ will present the manufacturer's code and $A_0 = V_{IH}$ the device code (see Table 1). When $A_9 = V_{IH}$, CE and OE default to active-low and ALE need not be toggled to latch each identifier address. This mode functions in the $25^\circ C \pm 5^\circ C$ ambient temperature range required during programming.

ERASURE CHARACTERISTICS (FOR Cerdip EPROMs)

Exposure to light of wavelength shorter than 4000 Angstroms (\AA) begins EPROM erasure. Sunlight and some fluorescent lamps have wavelengths in the 3000–4000 \AA range. Constant exposure to room-level fluorescent light can erase an EPROM in about 3 years (about 1 week for direct sunlight). Opaque labels over the window will prevent unintentional erasure under these lighting conditions.

The recommended erasure procedure is exposure to 2537 \AA ultraviolet light. The minimum integrated dose (intensity \times exposure time) is 15 Wsec/cm². Erasure time using a 12000 $\mu W/\text{cm}^2$ ultraviolet lamp is approximately 15 to 20 minutes. The EPROM should be placed about 1 inch from the lamp. The maximum integrated dose is 7258 Wsec/cm² (1 week @ 12000 $\mu W/\text{cm}^2$). High intensity UV light exposure for longer periods can cause permanent damage.

CHMOS NOISE CHARACTERISTICS

System reliability is enhanced by Intel's CHMOS EPI-process techniques. Protection on each data and address pin prevents latch-up; even with 100 mA currents and voltages from $-1V$ to $V_{CC} + 1V$. Additionally, the V_{PP} pin is designed to resist latch-up to the 14V maximum device limit.

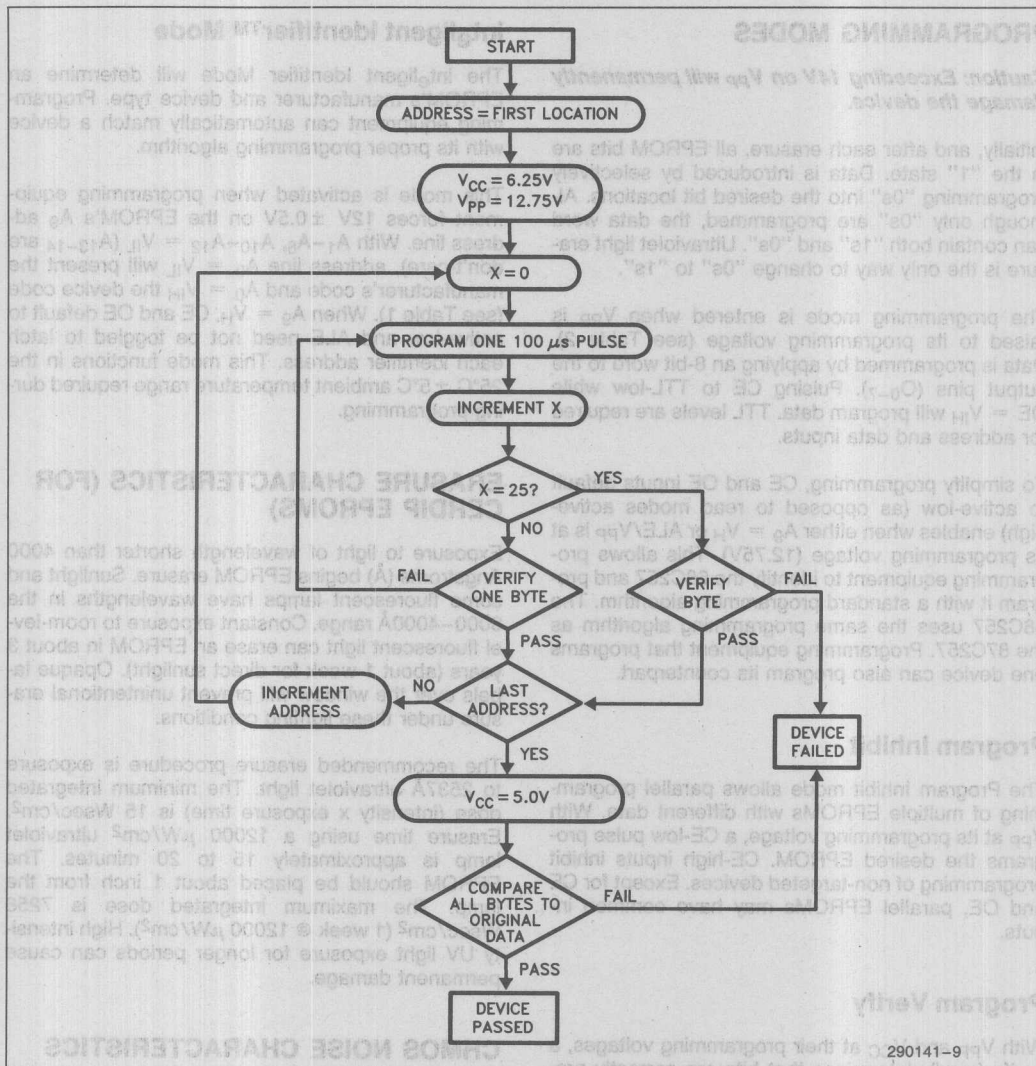


Figure 5. Quick-Pulse Programming™ Algorithm

Quick-Pulse Programming™ Algorithm

The Quick-Pulse Programming algorithm programs Intel's 68C257 EPROM. Developed to substantially reduce production programming throughput time, this algorithm can program a 68C257 in under four seconds. Actual programming time depends on the PROM programmer used.

The Quick-Pulse Programming algorithm uses a 100 microsecond initial-pulse followed by a byte verifica-

tion to determine when the addressed byte is correctly programmed. The algorithm terminates if 25 100μs pulses fail to program a byte. Figure 5 shows the Quick-Pulse Programming algorithm flowchart.

The entire program-pulse/byte-verify sequence is performed with $V_{CC} = 6.25V$ and $V_{PP} = 12.75V$. When programming is complete, all bytes should be compared to the original data with $V_{CC} = 5.0V$.

D.C. PROGRAMMING CHARACTERISTICS $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$

Table 2

Symbol	Parameter	Limits			Test Conditions
		Min	Max	Unit	
I_{LI}	Input Current (All Inputs)		1.0	μA	$V_{IN} = V_{IL} \text{ or } V_{IH}$
V_{IL}	Input Low Level (All Inputs)	-0.2	0.8	V	
V_{IH}	Input High Level	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage During Verify		0.4	V	$I_{OL} = 2.1 \text{ mA}$
V_{OH}	Output High Voltage During Verify	$V_{CC} - 0.8$		V	$I_{OH} = -400 \mu\text{A}$
$I_{CC2}^{(3)}$	V_{CC} Supply Current		30	mA	
$I_{PP2}^{(3)}$	V_{PP} Supply Current (Program)		50	mA	$CE = V_{IL}$
V_{ID}	A_9 intelligent Identifier Voltage	11.5	12.5	V	
$V_{PP}^{(1)}$	Programming Voltage	12.5	13.0	V	
$V_{CC}^{(1)}$	Supply Voltage During Programming	6.0	6.5	V	

A.C. PROGRAMMING CHARACTERISTICS

$T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$; see Table 2 for V_{CC} and V_{PP} voltages.

Symbol	Parameter	Limits				Conditions
		Min	Typ	Max	Unit	
t_{AS}	Address Setup Time	2			μs	
t_{OES}	OE Setup Time	2			μs	
t_{DS}	Data Setup Time	2			μs	
t_{AH}	Address Hold Time	0			μs	
t_{DH}	Data Hold Time	2			μs	
$t_{DFP}^{(2)}$	OE High to Output Float Delay	0		130	ns	
$t_{VPS}^{(1)}$	V_{PP} Setup Time	2			μs	
$t_{VCS}^{(1)}$	V_{CC} Setup Time	2			μs	
t_{PW}	CE Program Pulse Width	95	100	105	μs	
t_{OE}	Data Valid from OE			150	ns	

NOTES:

- V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP} .
- This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.
- The maximum current value is with outputs O_0 to O_7 unloaded.



NOTES:

1. The input timing reference level is $V_{IL} = 0.8V$ and $V_{IH} = 2V$.
2. t_{OE} and t_{DFP} are device characteristics but must be accommodated by the programmer.
3. To prevent device damage during programming, a $0.1 \mu F$ capacitor is required between V_{PP} and ground to suppress spurious voltage transients.
4. During programming, the address latch function is bypassed whenever $V_{PP} = 12.75V$ or $A_9 = V_H$. When V_{PP} and A_9 are at TTL levels, the address latch function is enabled, and the device functions in read mode.
5. V_{PP} can be $12.75V$ during Blank Check and Final Verify; if so, CE must be V_{IH} .

In-Vehicle Networking Background Information

7

In-Vehicle Networking Background Information

7

SAE The Engineering
Resource For
Advancing Mobility

400 COMMONWEALTH DRIVE WARRENDALE, PA 15096

Requirements and Directions

Frederick H. Phall
and David J. Arnett
Intel Corp.

SAE Technical Paper Series

860390

1.0 INTRODUCTION

In-Vehicle Networking — Serial Communication Requirements and Directions

Frederick H. Phall
and David J. Arnett

Intel Corp.

In recent years the functional requirements for in-vehicle networking have become more complex. The need for a more efficient, reliable, and cost-effective means of communication within the vehicle has become a reality. This paper discusses the requirements and directions for in-vehicle networking, focusing on serial communication. It addresses the challenges of integrating multiple electronic modules into a single network, the need for a common protocol, and the importance of reliability and security. The paper also discusses the current state of the art and provides a vision for the future of in-vehicle networking.

1.0 BACKGROUND

In-vehicle networking, often called multi-plexing, has been a discussion topic for years. It primarily addresses two areas - information sharing and body electronic control. Properly designed, the benefits which may be obtained are both large and measurable. The benefits may be measured in many areas of system cost, reduction and performance enhancement, including: enhanced diagnostics, reduction of sensor count, blackbox control, and cost.

- Open systems flexibility
 - High reliability in a noisy environment
 - Reduction of cost and manufacturing complexity
 - Minimum CPU burden for communication
 - Maximum program transparency
 - Guaranteed data consistency
 - Short transmission times
- International Congress and Exposition
Detroit, Michigan
February 24-28, 1986

270508-1

In-Vehicle Networking — Serial Communication Requirements and Directions

Frederick H. Phall
and David J. Arnett

Intel Corp.

ABSTRACT

In recent years the functionality of automotive systems has been improved by the introduction of real time Electronic Control Units (ECUs) for engine management, anti-lock braking, and other applications. For customer comfort and convenience, body electronics options have also increased, including electronic windows, seat control, and others. Optimization of performance requires integrating the vehicle of the 90s as a system rather than a grouping of individual modules. As a result, inter-communication between real time ECUs as well as between body electronics modules is required. By linking vehicle electronics into a network or combination of networks, a cost-effective solution which guarantees required performance and maximum flexibility may be obtained.

The following features are key requirements for distributed control and networking in automobiles:

- Open systems flexibility
- High reliability in a noisy environment
- Reduction of cost and manufacturing complexity
- Minimum CPU burden for communication
- Maximum programmer transparency
- Guaranteed data consistency
- Short transmission times

To date, no network system that meets all of these objectives has been available. This paper examines: in-vehicle networking applications; protocol features needed to meet requirements; the need for an automotive serial communication standard; and briefly shows how the protocol designed for in-vehicle communication, Inter-Controller Area Network, meets these needs.

1.0 INTRODUCTION

Automobile change is evolutionary. Two of the key factors in each design change are cost and performance. Component count or size reduction, reduced manufacturing complexity, and increased reliability (as well as fast and accurate repair) all reduce cost, making the car a more attractive purchase to the customer. Increased performance has the same impact. A smoother ride, higher fuel economy, and faster, more controlled response all increase customer satisfaction - i.e. "more bang for the buck". New vehicle designs and existing model changes are given the charter of accomplishing both of these objectives. The technology is present for the next evolutionary change. It allows the vehicle to be designed, tested and operated as a system. This technology is serial communication networking.

2.0 BACKGROUND

In-vehicle networking, often called multiplexing, has been a discussion topic for years. It primarily addresses two areas - information sharing and body electronics control. Properly designed, the benefits which may be obtained are both large and measurable. The benefits may be measured in many areas of system cost reduction and performance enhancement, including: enhanced diagnostics, reduction of sensor count, distributed control, and total wire reduction.

The term "multiplexing" is often used to describe many different things. For this paper it shall be used as defined by the SAE Multiplexing and Data Communications Subcommittee:

A system of transmitting several different messages on the same circuit or channel. [1]

In-vehicle networking has been integrated into several production programs already, in a number of different forms. As an example, General Motors Corporation, in their 1984-85 Buick E-family (Riviera) combined six separate modules and a diagnostic interface into a common network (see Figure 1) which uses both to broadcast and point-to-point communications [2].

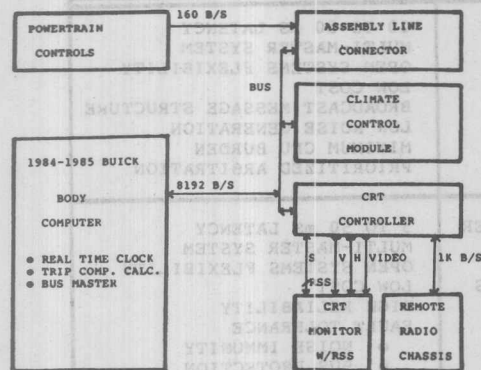


FIG. 1
1984 - 1985 BUICK E CRT DISPLAY SYSTEM USING
BOTH BROADCAST AND POINT-TO-POINT COMMUNICATIONS

Ford Motor Company, in the 1985 Mark VII Comtech program, integrated a CRT with 10 different vehicle functions (including climate control, fuel, and trip information) which may be scanned from a switch panel on the steering wheel and controlled on the touch screen [3]. These and other systems designs have provided the foundation for the development of optimized networks based on automotive requirements and objectives.

The requirements for in-vehicle networking communication and control are consistent (based on many automotive sources) but are dependent upon the network application. Network applications may be grouped into three key types (see Table 1): body electronics "smart-power" control, intermediate-speed information sharing; and high speed or real time control. These types have generally been labeled Types 1, 2, and 3. Type-1 for body electronics "smart-power" control (lights, power locks, etc.), Type-2 for status and non-critical information sharing (display, engine, etc.), and Type-3 for real-time control (powertrain, chassis, etc.). The application types have a number of requirements in common, such as: high reliability, open systems flexibility, and diagnosis capability. However, several requirements are dependent on type, such as: data rate, latency, and acceptable burden on the control CPU.

Based on the above directions and requirements (of which only a few were mentioned), there is presently no commercially available serial communication protocol or protocol

implementation which adequately meets and serves automotive needs. The following sections detail: Type-1, Type-2, and Type-3 directions; the requirements for each and ways in which they may be met; the basis on which an automotive protocol must be defined; and the Inter-Controller Area Network proposal. Because Type-3 networks have the highest performance applications, and the most strict requirements, they will be discussed first, followed by Type-2 and Type-1.

3.0 TYPE-3 DIRECTIONS AND REQUIREMENTS

Type-3, or real-time control applications receive the most exposure publicly, by their nature, and have the highest performance requirements. While both Type-3 and Type-2 applications perform information sharing, they differ in that Type-3 passes information critical for control (see Figure 2) and Type-2 passes information primarily for status (see Figure 3) and display purposes. As an example of a Type-3 application, consider a distributed powertrain control application where the fuel and spark rate are being controlled separately. To prevent redundant sensors, only one of these modules needs to input engine speed, and that module has the responsibility to convert it to an easily understandable format and broadcast it to other modules requiring it. Typically, based on program loops, the other controllers must have this data within a maximum latency time of 5 milliseconds.

In Type-3 networks the system design is critical. If the network and system are designed properly, performance may be optimized, interactive diagnostics effectively introduced, and redundancies removed. In the same way, if a system is not designed properly, damage may occur before diagnostics are invoked or the driver is alerted.

Networking communication between electronic controls in the vehicle allows many system improvements. The objective is increased performance and reduced cost. Type-3 applications achieve this in networking primarily through enhanced diagnostics, distributed control, algorithm optimization, and sensor reduction. An overview of the benefits obtained in these areas follows.

3.1 TYPE-3 BENEFITS TO SYSTEM

Enhanced diagnostics is perhaps the driving factor for bringing inter-module communication into the vehicle. As more electronics are integrated into vehicles, finding and correcting problems becomes increasingly difficult. In a networked environment, a mechanic may electronically request information and "last status" through the communication bus, and determine where the failure is and what correction is required.

TABLE 1
NETWORK APPLICATIONS

APPLICATION	TYPICAL NODES IN NETWORK	REQUIREMENTS AND FEATURES NEEDED
TYPE-1 (BODY-ELECTRONICS CONTROL)	POWER LOCKS POWER SEATS POWER WINDOWS POWER MIRRORS LAMPS CLIMATE CONTROL ENTERTAINMENT	20 TO 50 ms LATENCY MULTI-MASTER SYSTEM OPEN SYSTEMS FLEXIBILITY LOW COST BROADCAST MESSAGE STRUCTURE LOW NOISE GENERATION MINIMUM CPU BURDEN PRIORITIZED ARBITRATION
TYPE-2 (STATUS-TYPE INFORMATION SHARING)	INSTRUMENT CLUSTER BODY COMPUTER ENGINE STATUS SUSPENSION STATUS ANTI-LOCK STATUS NAVIGATION TRIP COMPUTER DIAGNOSTICS CRT	5 TO 50 ms LATENCY MULTI-MASTER SYSTEM OPEN SYSTEMS FLEXIBILITY LOW COST HIGH RELIABILITY FAULT TOLERANCE <ul style="list-style-type: none"> ● NOISE IMMUNITY ● BUS PROTECTION BROADCAST MESSAGE STRUCTURE MINIMUM CPU BURDEN PRIORITIZED ARBITRATION
TYPE-3 (REAL-TIME CONTROL)	CENTRAL POWERTRAIN FUEL CONTROL SPARK CONTROL TRANS. CONTROL ANTI-LOCK SUSPENSION DIAGNOSTICS DRIVER AIDS	1 TO 5 ms LATENCY MULTI-MASTER SYSTEM OPEN SYSTEMS FLEXIBILITY HIGH RELIABILITY FAULT TOLERANCE <ul style="list-style-type: none"> ● NOISE IMMUNITY ● BUS PROTECTION BROADCAST MESSAGE STRUCTURE MINIMUM CPU BURDEN PRIORITIZED ARBITRATION DATA CONSISTENCY

Through the same communication mechanism, the driver may be alerted to failures or potential failures through information sent to the display. Thus driver awareness is increased and the extent of potential damage from the failure is reduced.

Distributed control is another design evolution which is made possible through in-vehicle networking. U.S. automotive manufacturers primarily have been and are continuing to design integrated systems, while Europe and Japan are designing more distributed control networks [4]. It is not within the scope of this paper to discuss the trade-offs of either approach. For distributed control however, high speed serial communication networking enhances sharing of common data, higher performance algorithm development, and greater fault tolerance.

CPU availability is one of the most precious and important factors in a control

system. Higher CPU throughput allows more complex and/or better control, and a more competitive product. Through information or parameter passing, all the nodes in the network have access to information such as a sensor inputs "hot off the press" or the results of extensive calculations, such as engine speed or a stoichiometric load variable. The nodes receiving and using such information do not have to go through the same computation themselves - allowing them to spend more time and computation power on other important calculations or control. In addition, they are guaranteed data consistency for inter-related control applications. Integrating control modules onto a network should not decrease performance by increasing CPU burden.

Sensor reduction is possible through information sharing. The sensor input may be stored by its primary user and then distributed to other controllers requiring the information.

270508-4

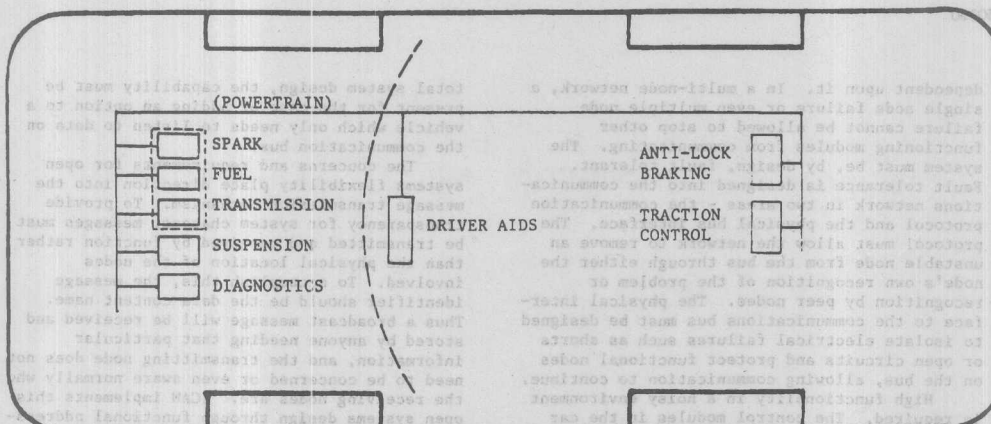


FIGURE 2: TYPE III APPLICATION (REAL-TIME CONTROL)

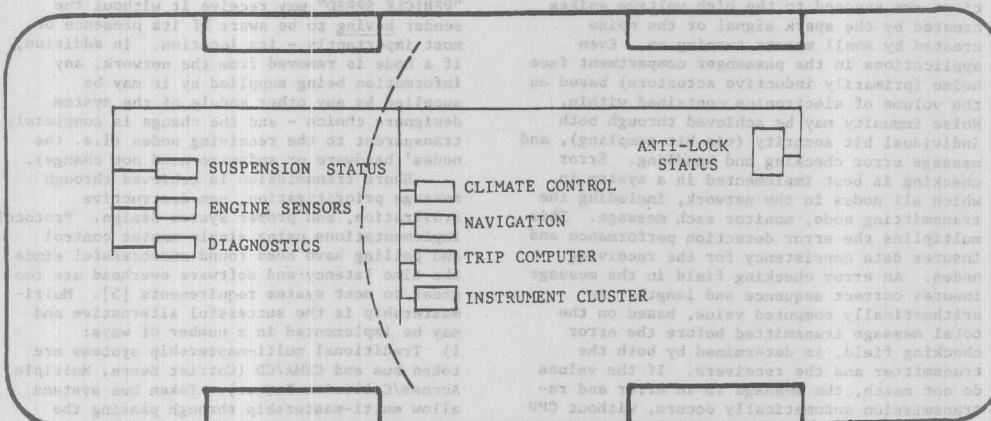


FIGURE 3: TYPE II APPLICATION (STATUS INFORMATION SHARING)

This results in a sensor reduction but, perhaps more importantly, guarantees data consistency. The designer is insured that nodes using the same input information (engine speed, coolant temperature) will use data sensed at the same time.

3.0 TYPE-3 NETWORK REQUIREMENTS

The network performance requirements to be met for Type-3 (real-time control) networks are extensive and demanding. Based on automotive engineering inputs, the key requirements in priority are:

- High reliability
- High functionality in a noisy environment
- Open system flexibility

- Short transmission delay for urgent messages
 - Maximum CPU and programmer transparency
- To meet these requirements, the following key features are required.
- High fault tolerance/isolation
 - Noise immunity, error checking
 - Broadcast message structure
 - Multi-mastership, contention
 - Prioritized messages, with bitwise arbitration
 - High data rate
 - Variable length data fields
 - Intelligent, transparent message storage
- High reliability is the highest priority requirement since system performance and vehicle "extent of damage" are directly

270508-5

860390

dependent upon it. In a multi-node network, a single node failure or even multiple node failure cannot be allowed to stop other functioning modules from communicating. The system must be, by design, fault tolerant. Fault tolerance is designed into the communications network in two areas - the communication protocol and the physical bus interface. The protocol must allow the network to remove an unstable node from the bus through either the node's own recognition of the problem or recognition by peer nodes. The physical interface to the communications bus must be designed to isolate electrical failures such as shorts or open circuits and protect functional nodes on the bus, allowing communication to continue.

High functionality in a noisy environment is required. The control modules in the car are exposed to an extremely harsh environment in terms of temperature, humidity, and electrical noise. Under-hood control applications are exposed to the high voltage spikes created by the spark signal or the noise created by small motors turning on. Even applications in the passenger compartment face noise (primarily inductive actuators) based on the volume of electronics contained within. Noise immunity may be achieved through both individual bit security (via bit sampling), and message error checking and handling. Error checking is best implemented in a system in which all nodes in the network, including the transmitting node, monitor each message. This multiplies the error detection performance and insures data consistency for the receiving nodes. An error checking field in the message insures correct sequence and length. An arithmetically computed value, based on the total message transmitted before the error checking field, is determined by both the transmitter and the receivers. If the values do not match, the message is in error and retransmission automatically occurs, without CPU intervention. The Inter-Controller Area Network (ICAN) protocol (see Section 7.0 for details) accomplishes each of the above tasks. All nodes monitor each transmission for errors, and a 16-bit Cyclic Redundancy Check (CRC) is calculated for each message and used to confirm the message accuracy.

Open systems flexibility allows "transparent" network expansion or reduction. The result of open systems design is that the addition of a module does not, unless it is desired, impact existing modules. Subtraction of a module does not impact existing unrelated modules. Vehicle designs change. Addition or subtraction of a module in the factory or at a dealership should absolutely not require hardware or software modifications in other areas in the vehicle. There are exceptions however, such as adding a node which provides new information which existing nodes wish to use. An algorithm change is then required for those receiving nodes, regardless. However, for the

total system design, the capability must be present for the case of adding an option to a vehicle which only needs to listen to data on the communication bus.

The concerns and requirements for open systems flexibility place direction into the message transmission mechanism. To provide transparency for system changes, messages must be transmitted and received by function rather than the physical location of the nodes involved. To accomplish this, the message identifier should be the data content name. Thus a broadcast message will be received and stored by anyone needing that particular information, and the transmitting node does not need to be concerned or even aware normally who the receiving nodes are. ICAN implements this open systems design through functional addressing. Messages are transmitted, and received, based on their content - rather than physical addresses. Hence, any node wishing to receive "VEHICLE SPEED" may receive it without the sender having to be aware of its presence or most importantly - its location. In addition, if a node is removed from the network, any information being supplied by it may be supplied by any other module of the system designers choice - and the change is completely transparent to the receiving nodes (i.e. the nodes' hardware or software need not change).

Short transmission is achieved through message prioritization, non-destructive arbitration, and proper system design. Protocol implementations using single-master control and polling have been found unsuccessful since the time latency and software overhead are too great to meet system requirements [5]. Multi-mastership is the successful alternative and may be implemented in a number of ways:

- 1) Traditional multi-mastership systems are token bus and CSMA/CD (Carrier Sense, Multiple Access/Collision Detect). Token bus systems allow multi-mastership through passing the mastership with a "token", a special message or electrical signal [6]. While token bus designs have merit, they suffer in that only one node may have mastership at a time and substantial time and logic is required to recover the system if a node goes down, especially if the node has the token.
- 2) The CSMA/CD protocol allows simultaneous mastership, where a node may transmit any time the bus is free. The drawback in the traditional approach however, is that if two nodes begin transmitting at the same time the message will be destroyed and each node will back off the bus for a random time. This creates substantial time loss and the need for increased timer logic in the system.
- 3) A successful alternative may use bitwise arbitration on a contention bus, allowing non-destructive collisions - i.e. the highest priority wins. The physical bus has a dominant logic state (often "0"), and if two nodes begin transmitting simultaneously, the node which

270508-6

transmits a recessive bit first will lose the arbitration. This happens because the other node transmitted a dominant bit in the same bit time and wrote "over" it. By monitoring the bus during transmission, the losing node then backs off and becomes a receiver. To achieve maximum performance, the priority level assigned should be integrated into the identifier field. The result is that the highest priority messages are assigned the highest priority names, guaranteeing that they will win arbitration and transmit with minimum latency. ICAN implements bitwise arbitration as described above. Hence the bus is always used at 100 percent efficiency - even when collisions occur. By placing the priority in the identifier field, ICAN also reduces the number of bits per message, reducing the bus load and the latency time.

To be accepted, the network should not reduce the performance of the existing controllers in the vehicle. The controller CPUs are often near their throughput limit and cannot be required to dedicate substantial time to controlling the communication. Secondly, system designers and software programmers should not be required to know the complete logical operation of the communication interface, since engineering development time for production programs is valuable and short. In the ICAN design the communication interface is buffered such that the programmer and CPU need only store, in their own RAM, information to be transferred to other controllers. The on-chip interface logic then transfers this new information from the RAM to the bus, completely transparent to the CPU. The received messages are handled in the same manner. As a result, all framing, serialization, error detection, and retransmission are performed transparent to the CPU. The CPU is alerted however, when involvement is required for failure diagnosis, such as when a node or series of nodes fail (and system or algorithm modification is required).

3.3 TYPE-3 APPLICATION TRENDS AND TIME FRAMES

Likely network designs will primarily link four major areas: powertrain, chassis, diagnostics and driver aids. Powertrain electronics may be distributed either within a physical module (box) or between modules. The functional partitioning and information passed will probably be the same, regardless of physical partitioning. Powertrain areas include control functions such as spark, fuel, and transmission. Chassis electronics include control of: suspension, anti-lock braking, traction, and collision prevention. Diagnostics includes both monitoring responsibilities (like data logging of failures), and interactive control (to protect and modify the system in failure situations and allow the driver to get home). Driver aids are primarily a Type-2 function, but are integrated in Type-3 to alert the driver of potential failures and/or danger.

Type-3 networks will probably be the last chronologically, of the three applications to reach production in U.S. programs. This is primarily due to cost and perceived need. The high bus rates needed by Type-3 networks require more costly medias - (and their respective drivers) - such as coax, twin shielded pair, or fiber optics. In the case of fiber optics the technology is not completely present (a low loss "T" connection has not been proven). The perceived need has not reached the required level yet because Type-3 networks are an optimization rather than a solution to an immediate problem.

4.0 TYPE-2 DIRECTIONS AND REQUIREMENTS

Type-2 applications provide status and display data communications between intelligent modules, compared to Type-3 applications which pass distributed control information. Potential systems may network modules or functions such as: entertainment, displays, climate control, engine status, trip computer, etc. The primary functions of Type-2 networks are to pass important but not critical information. While Type-3 applications are the most demanding, and Type-1 applications have been the most frequently discussed, Type-2 networks will likely see production first on a broad base, because they offer the highest immediate benefit.

4.1 TYPE-2 BENEFITS TO SYSTEM

The benefits that can be obtained in a Type-2 network are similar to those of Type-1 but more easily measured. They include:

- reduced sensor count
- increased diagnostics
- reduction in manufacturing complexity
- distributed modules
- increased driver awareness

Reduced sensor count is an immediately measurable benefit. Electronic displays are rapidly becoming the norm, and are chartered with displaying as much information as the driver can understand, with as much clarity as possible. The displayed information comes from many functional systems in the vehicle, such as powertrain, chassis, and entertainment. To show the information, the display may have either: independent sensors for all the information, dedicated signal lines to each functional system, or a common bus which links the modules together. Networking implements the latter. The primary user of an input is typically assigned the dedicated sensor and is responsible for broadcasting the data to other nodes requiring it. As a result, the display may obtain its data without having a full complement of sensors, and at the same time be assured that its data is the same being used by related control modules.

860390

Increased diagnostics may be achieved in three areas. First, failures may be rapidly and accurately diagnosed by a technician. Second, interactive "running" diagnostics may help the system to function adequately to get the driver to a repair stations. And third, factory automation is enhanced by providing the capability for individual systems to be tested before being placed in the vehicle or leaving the factory. An example of the latter is General Motors' Assembly Line Diagnostic Link (ALDL) which allows the manufacturer to perform diagnostic tests on the vehicle before it leaves the assembly line [2]. This reduces field failures and thus increases quality while reducing long term warranty overhead.

Distributed systems may be more easily designed and give much higher reliability with proper networking. This attribute does not differ from the discussion of Type-1 applications. The primary difference may be found in the benefit. A large portion of Type-2 functions are "options". Thus, a building block approach may be designed which gives the customer what he desires and can reduce his, as well as the manufacturer's, price appropriately. As an example, electronic instrument clusters cover a broad spectrum in features and performance. Typically, the modules have been internally generic since all the information was required for a superset display. Entry-level displays using the same electronics cost both the manufacturer and the customer more per feature. Using distributed information networking, the displays may be designed as building blocks to the superset; thus entry level displays obtain a more proportional cost.

Directly related to the displays is increased driver awareness. While real-time control information may be difficult for the driver to interpret and monitor, status information such as oil pressure, engine temperature, and fuel level are very important. From this, the driver may quickly be alerted to failure conditions and respond appropriately.

4.2 TYPE-2 NETWORK REQUIREMENTS

Type-2 network requirements, for the most part, are as extensive but not as demanding as Type-3 requirements. As an example, acceptable data latency is based on human perception rather than mechanical response. Thus the latency time can extend from the Type-3 requirement of 5 milliseconds up to as high as 50 milliseconds. System cost plays an important role and Type-2 networks are much more cost conscious relatively than Type-3. Briefly, the overall requirements include:

- Open systems flexibility
- Intermediate, but predictable latency
- Minimal CPU burden
- Low cost
- No measurable noise generated in AM or FM band
- Wide product offering

Open systems flexibility and acceptable, predictable latency have been discussed for Type-3 networks. The same general requirements hold true here. The system must be designed such that the addition or subtraction of a module does not necessarily require that existing modules in the vehicle require modifications. ICAN resolves this problem through functional messages - messages which are sent based on content and received based on content. The acceptable latency for Type-2 systems is an order of magnitude greater than Type-1 systems. The information is display-based or status-based, and as such need only be updated faster than human perception. An example of this is the electronic tachometer. When the driver accelerates sharply, he expects to see an "almost immediate" increase in the tachometer display. While 50 milliseconds delay would go unnoticed, 100 milliseconds may cause him concern about his engine or powertrain response. Processing requirements are not as stringent for Type-2 applications as for Type-1, but this allows lower cost microcontrollers to be used. Adding the control module to the network requires an enhancement either a more powerful processor to handle the interface in software, or dedicated logic to allow the system designer to continue using his existing design. The latter enhancement has some merit - the designer need not re-write his application software and may link to the network by simply storing and receiving information from the RAM array.

It is important at this point to understand several considerations. First, a contention network has been consistently chosen for this application because it performs well with random changes, with broadcast messages, and allows the most efficient use of the bus. HOWEVER, the traditional contention network is not acceptable for the following reasons:

- a) Destructive collisions with random back-off schemes remove any means to predict performance, and heavily load the bus.
- b) Lack of arbitration in the case of collisions removes the capability for prioritization - important messages have no way of being assured of bus access within their acceptable latency.

For these, and other requirements which will be discussed later, ICAN was designed - and performs as a true automotive protocol. Through hardware logic, ICAN performs bitwise arbitration so that in the case of a collision (a) one message will win and continue to transmit, and (b) the winner will be the highest priority of the colliding messages. Standard serial interfaces of existing microcontrollers do not allow this capability. The arbitration in today's devices would be done destructively. In addition, error checking would not be completed until long after the message is complete, raising a number of complications.

270508-8

To support the needed capability then, new logic must be developed and supported on a wide product base. To meet automotive manufacturers requirements, this product base should include both 8 and 16 bit CPUs, on industry standard architectures. This takes place in two ways: (a) the logic is integrated into the same microcontroller architectures being used now, which provides the highest performance and most transparent interface, and (b) a stand-alone peripheral is required to allow microcontroller architectures not having (at that time) the new serial interface to be linked to the network. (Planned products and directions will be discussed in later papers.)

One of the primary concerns in Type-2 applications is the electrical noise which could be generated by waveforms transmitted on the communication bus. Type-2 applications, for manufacturing and cost reasons, would like to use as little shielded media as possible and as a result are very sensitive to noise. Radio Frequency Interference (RFI) is unattractive to customers if noticeable, and thus unacceptable to automotive manufacturers. As a result, extensive research is being conducted by both automotive manufacturers and semiconductor suppliers to examine the trade-offs between bus medias, bus rates, and bus drivers/tranceivers.

4.3 TYPE-2 APPLICATION TRENDS AND TIME FRAMES

The implementation of networks has already begun. Many automotive manufacturers have either designed or are in the process of designing in-vehicle networks using proprietary protocols. In the United States, automotive manufacturers already have production programs or will have in the near future (including GM's E-K programs [2] and Ford's Comtech program [3]). European automotive manufacturers have begun in-vehicle electronic development later

than U.S. manufacturers (due to emission and mileage constraint differences) but have made significant progress and will be very competitive in this area.

5.0 TYPE-1 DIRECTIONS AND REQUIREMENTS

Type-1 applications are responsible for much of the early discussion of multiplexing, and hold high implementation potential. As shown in the examples in Figure 4, Type-1 applications network what are commonly called "body electronics". Body electronics encompass non-real-time, non-"big box" controls as a rule, meaning that body electronics includes: lamps, locks, windows, seats, displays, pod controls, etc. Many of the controls are switch inputs and outputs, being command controlled such as: "PARKING LIGHTS ON", "DOOR LOCKS OFF", "WINDOW DOWN". However, small motors are also encompassed for controls such as "MOVE SEAT BACK UNTIL...".

Type-1 system changes are much more random than Type-2 or Type-3 since they are usually initiated by the driver or passengers. Type-1 messages are command type, as described above. Since Type-1 actions are driver initiated and monitored, message latency and system response time are dependent on driver perception. Based on automotive research, human perception to a change is not less than 50 milliseconds. Based on relative system cost, Type-1 networks are much more cost-conscious than Type-2 or Type-3 [7]. However, for optimum vehicle performance and diagnostics, a standard protocol which meets the needs of both networks should be used.

5.1 TYPE-1 BENEFITS TO SYSTEM

There are four primary benefits in Type-1 networking:

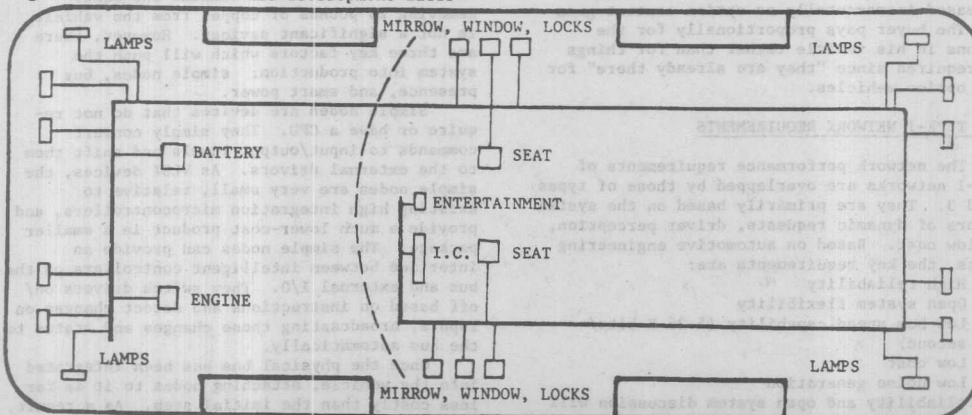


FIGURE 4: TYPE I APPLICATION (BODY ELECTRONICS CONTROL)

860390

- Increased diagnostics
- Signal wire reduction
- Migration of cost to individual functions
- Greater reliability and fault tolerance

The benefits achieved in increased diagnostics and greater reliability and fault tolerance, as well as the mechanism for which they may be achieved, were detailed during the Type-3 discussion. An overview follows, which describes the benefits obtained in signal wire reduction and migration of cost.

The amount of body electronics in automobiles has grown rapidly. Features such as electronic seat control, power windows and locks, remote radio, and keyless entry have brought significantly higher wiring content and complexity. While wire cost is relatively small, the manufacturing impact of increased wire is substantial. Increased wire content consumes larger space (now considered precious and difficult to find) in the vehicle and extends the amount of labor required for installation. Through serial communication, multiplexing reduces a large bundle of wires down to two or four. Based on the data rates required, the bus media may be twisted or even untwisted pair.

Networking the body electronics through multiplexing is perceived as the correct direction by many. However, there are a number of considerations, focusing primarily on cost. To be cost effective, a design should either reduce cost or be "value-added" and preferably should achieve both. Diagnostics and reliability, which are difficult to measure in dollars, provide the value-added benefit; while reduced manufacturing complexity and cost migration to functions provide the cost reductions. Through distributed intelligence using either zone controllers or smart power (as the technology grows), individual options take on the burden of their control electronics. The result is a "building block" design through which cost is increased incrementally as option content goes up. The buyer pays proportionally for the options in his vehicle rather than for things not required since "they are already there" for high option vehicles.

5.2 TYPE-1 NETWORK REQUIREMENTS

The network performance requirements of Type-1 networks are overlapped by those of types 2 and 3. They are primarily based on the system factors of dynamic requests, driver perception, and low cost. Based on automotive engineering inputs, the key requirements are:

- High reliability
- Open system flexibility
- Low bus speed capability (5-25 K bits/second)
- Low cost
- Low noise generation

The reliability and open system discussion will not be repeated here. However, it is important to note that the selected bus speed is typically

the lowest speed that will still guarantee the maximum latency. As an example, when the driver touches the lock switch, he expects "immediate" response. If this does not occur within his perception of immediate, he will touch the switch again, thinking that it was not correctly initiated the first time. This, on a repetitive basis, causes customer frustration and/or irritation and is unacceptable.

Cost is a very key factor in Type-1 implementations. However, if the "go/no-go" decision is based purely on saving current dollars, systems usually won't reach production. While the bulk of the wire may be reduced, the quantity of cut leads (where cost is determined) will not be reduced. The real cost savings is in manufacturing time and complexity, and in the lifetime cost for vehicle diagnostics.

As an example of wire bulk reduction, there are 52 wires leading into the door of a Cadillac [8]. By multiplexing, this could be reduced to four wires. Not only does this simplify installation of the wiring harness, it enhances total assembly. Vehicles are often assembled in a "doors off" technique, in which the doors are assembled and instrumented separately from the passenger compartment, then joined later in the assembly line. Because of the bulk of wires leading from the passenger compartment to the door, present techniques require the enlargement of the connectors to the point where future programs would become virtually impossible.

5.3 TYPE-1 APPLICATION TRENDS AND TIME FRAMES

Type-1 applications have obtained a great deal of exposure, and in many areas are perceived as a positive and necessary evolution. The primary area limiting their growth is the cost trade-off. Bulk wire is considered "free" in comparison to other system factors, since the number of cut leads remains the same. Removing 25 pounds of copper from the vehicle is not a significant savings. However, there are three key factors which will push the system into production: simple nodes, bus presence, and smart power.

Simple nodes are devices that do not require or have a CPU. They simply convert commands to input/output levels and shift them to the external drivers. As VLSI devices, the simple nodes are very small, relative to existing high integration microcontrollers, and provide a much lower-cost product in a smaller package. The simple nodes can provide an interface between intelligent controllers on the bus and external I/O. They switch drivers on/off based on instructions and detect changes on inputs, broadcasting those changes and status to the bus automatically.

Once the physical bus has been integrated into the vehicle, attaching nodes to it is far less costly than the initial step. As a result, Type-1 applications may very well make their appearance in volume once the Type-2 system has

270508-10

already been placed in the vehicle, using the same bus.

"Smart power" devices are components on which low voltage signals (input) can control high voltages and currents (output), on the same component. Because the two process technologies are traditionally separate and incompatible, they have not had a significant impact yet on the market. As "smart power" technology matures, Type-1 designs will expand. The component reduction and easier packaging capability will allow more distributed capability. Regardless of time and technology though, Type-1 systems solve very real manufacturing problems and production programs are being implemented.

6.0 SUMMARY OF APPLICATION TYPES AND IMPACT

In-vehicle networking may be separated functionally, and perhaps physically, into three areas: real-time control, status-type data sharing, and body electronics control. While each of the areas addresses different applications and has different specific requirements, the basic needs are the same. Needs which are common include:

- Reliability
- Functionality in a noisy environment
- Open system flexibility
- Low noise generation
- Minimal CPU burden
- Variable length data fields
- Programmer transparency

Specific requirements which are not common include:

- Maximum data latency
- Bus data rate
- Required data consistency

Although in-vehicle networking applications really have many of the same requirements, the differences lie primarily in the rate at which the actions occur. Based on growing vehicle needs, networking will reach - and in many cases already has reached - production, and the networks will eventually encompass all three applications. The applications may or may not be connected on the same physical bus, but there will be a need to communicate between the three application types.

To effectively network the vehicle and allow inter-communication between the application types, one positive approach is to implement a common protocol throughout the vehicle. This allows straightforward communication between the different application buses - and more importantly allows one diagnostic tool to service the vehicle, compared to possibly three tools to service three protocols in the same vehicle. The protocol needs to be able to meet the above needs and function effectively over a broad data rate range. The protocol, as a result, must be cost conscious for the Type-1 applications, but not limit the Type-3 applications in performance.

To determine the acceptable cost trade-offs it is necessary to examine the total system cost for each alternative, in addition to the component cost. A perceived advantage of having different protocols for each application in the vehicle is:

- Reduced silicon overhead for lower type applications

Which must be compared to the perceived disadvantages of:

- Multiple diagnostic interfaces
- Comparative complexity of gateways
- System engineer's knowledge requirements - understanding three protocols
- Increased supplier support requirements

These issues are by no means straightforward, and must be carefully evaluated. It is not within the scope of this paper to perform this evaluation. Based on the direction that a common protocol is required, however, the Inter-Controller Area Network (ICAN) has been developed to meet all of the above needs. It is briefly discussed in the following section.

7.0 THE INTER-CONTROLLER AREA NETWORK - A BRIEF SUMMARY

The Inter-Controller Area Network (ICAN) is serial communication protocol designed specifically for in-vehicle networking. It provides a powerful means for data communication and control information transfer. The ICAN principle is to support a global network of multimaster nodes consisting of the control CPU (if the application requires it) and the ICAN interface. The CPU controls the attached process by its I/O lines; transfer of messages between different nodes is initiated by the CPU but actually executed by the ICAN interface. The ICAN interface may be integrated onto the microcontroller or a separate peripheral. The interface consists of buffer memory and an intelligent bus interface.

The protocol is a contention-based (multi-master) protocol which performs non-destructive bitwise arbitration and has prioritized messages. Transmission and reception of messages is performed transparently to the CPU, and error detection and correction is performed autonomously by each bus interface.

Messages are broadcast to the network, for access by any node needing the data. Messages are identified (transmitted and received) by content, and prioritization of the messages is defined by the system designer.

Error checking is very advanced through the 16 bit Cyclic Redundancy Check (CRC) and a number of embedded features. Error correction is performed through automatic re-transmission and error is confined in the case of local node failure.

System recovery is easily and quickly done from scratch and restart performs fluently independent of the system's configuration.

860390

ICAN has been submitted, and is further detailed, in the proposed information report to the SAE Vehicle Network for Multiplexing and Data Communications (multiplex) subcommittee. The proposed information report may be referenced through SAE number J1583 [9].

8.0 THE NEED FOR A STANDARD

An industry standard is absolutely required. As work and research are done on different protocols and implementation by independent engineering teams from automotive manufacturers and semiconductor suppliers one direction becomes very clear - the needs and requirements are the same. On the basis that one protocol can meet the needs of different automotive manufacturers, there are many things to be gained by a common standard:

- Common diagnostic tools and support
- Reduced component design engineering time
- Enhanced supplier support and innovation
- Enhanced manufacturing - reduced complexity
- Fluent evolution of future in-vehicle networks

One of the primary objectives for in-vehicle networking is increased diagnostic capability. The service mechanic should be able to locate and repair or replace the failure quickly. To accomplish this, diagnostic tool consistency is important. The same tool should be able to communicate directly on the physical bus and request information of perform other diagnostics for a number of model year vehicles, for a number of manufacturers. This does not preclude competitive design. Different vehicle models and manufacturers will have different modules in the system and use different message identifiers. However, one tool may still communicate on the bus, with a ROM or other interface that provides the message identifiers for that vehicle. Vehicle information may be provided through means anywhere from the vehicle itself to a phone link to the vehicle manufacturer.

The design of multiple VLSI components to perform multiple protocols impacts both the semiconductor supplier and the automotive manufacturer. For the semiconductor supplier, it consumes extensive design and support resources and extends development time. For the automotive manufacturer, it extends the time before components are available for production, reduces possible price reduction by volume, and perhaps most importantly, reduces the competitiveness and design innovations of multiple semiconductor suppliers. An in-vehicle standard allows the semiconductor suppliers to support a protocol across their automotive product line, to enhance its performance and increase its capability, and provide a cost-conscious product.

A standard allows vehicle manufacturing to play an integral and consistent role in testing and assembly. The capability to test modules before they are placed into a vehicle and then as a system in the assembled vehicle has merit. As

stated before, GM currently does this with the Assembly Line Diagnostic Link (ALDL), but much higher capability is desired for the future. As factory automation via common protocol becomes a mature method, a gateway interface to a vehicle which speaks one "language" provides increased manufacturing performance and consistency. It is important to note however, that the protocols for manufacturing and for in-vehicle communication will not be the same - the needs and end designs are radically different. A fluent gateway or database must be provided between them.

Much frustration has arisen from consistently changing in-vehicle protocols. Large engineering efforts by automotive engineering teams are required to support a network design; and changing the protocol used, in short time intervals, does not take advantage of that effort. A standard gives a constant and planned direction. The direction allows consistent support for future designs by both manufacturers and suppliers.

9.0 SUMMARY

In-vehicle electronics have expanded and provide higher functionality and increased control. Optimization of performance requires integrating the vehicle of the 90s as a system rather than a grouping of individual modules. In-vehicle networking may be partitioned into three areas: body electronics control, status data sharing, and real-time control; referred to as Type-1, Type-2 and Type-3 respectively. The applications all offer performance and long-term cost advantages when networked and primarily have common requirements. No existing protocol meets all of the application type requirements. As a result ICAN has been developed. ICAN is a contention-based, high-performance protocol that operates over a broad speed range and uses bitwise prioritized arbitration with functional addressing. A standard protocol is needed, to enhance diagnostic and manufacturing capability. The standard must be supported by automotive manufacturers and semiconductor suppliers.

REFERENCES

- [1] Glossary of Vehicle Networks for Multiplexing and Data Communications
- [2] Ronald W. Cox, Delco Electronics Division of General Motors Corporation, "Local Area Network Technology Applied to Automotive Electronic Communications", October 25, 1985, IEEE AUTOMOTIVE APPLICATIONS OF MICROPROCESSORS
- [3] Ronald K. Jurgen, Senior Editor, "More electronics in Detroit's 1985 models", October, 1984, IEEE Spectrum

270508-12

[4]. Erik L. Keller, Editor, "Automotive Electronics Shifts Into High Gear", January 26, 1985, ELECTRONICS

[5]. Boyd Nichols, John Deere Product Engineering Center, "A DATA LINK FOR AGRICULTURAL AND OFF-HIGHWAY COMMUNICATIONS", October 25, 1985, IEEE AUTOMOTIVE APPLICATIONS OF MICROPROCESSORS

[6]. MAP V2.1 Specification

[7]. Shigeru OHO and Takao Sasayama, Hitachi Research Laboratory, "A Custom LSI Approach To Multiplexed Wiring", ISATA 85046

[8]. Herb Brody, Senior Editor, "SMART POWER", December, 1985, High Technology

[9]. W. Lawrenz, Intel Corporation, "ICAN - INTER-CONTROLLER AREA NETWORK FOR IN-VEHICLE APPLICATIONS", Proposed Information Report J1583 for SAE Multiplex Subcommittee

[10]. Frederick Miesterfeld, Chrysler, "Chrysler Collision Detection (C2D) Serial Data Communications Multiplex Bus", Proposed Information Report for SAE Multiplex Subcommittee

[11]. Jack R. Volk and Wayne J. Johnson, Ford Motor Company, "Proposal for a Vehicle Network Protocol (VNP)", September 24, 1985, Proposed Information Report for SAE Multiplex Subcommittee

270508-13

SAE The Engineering Society
For Advancing Mobility
Land Sea Air and Space

400 COMMONWEALTH DRIVE WARRENDALE, PA 15096

SAE Technical Paper Series

870823

A High Performance Solution for In-Vehicle Networking — 'Controller Area Network (CAN)'

David J. Arnett

Intel Corp.

Earthmoving Industry Conference
Peoria, Illinois
April 7-9, 1987

270509-1

A High Performance Solution for In-Vehicle Networking — 'Controller Area Network (CAN)'

David J. Arnett

Intel Corp.

ABSTRACT

Recently, significant focus and development effort has been dedicated toward in-vehicle networking. This effort includes work on behalf of the American Trucking Association (ATA), the Society of Automotive Engineers (SAE), the International Standards Organization (ISO), and independent developments by automotive and semiconductor manufacturers. In-vehicle networking extends, as a result, beyond passenger cars into heavy truck, military, and construction vehicles. In the course of these developments, the benefits of networking have been examined and networking is perceived as having significant benefits, resulting in production and custom development [1,2,3].

The Controller Area Network (CAN) is a high-performance serial communication solution which has been designed to meet the requirements for the broad range of applications and has now progressed from a specification to a product.

This paper will detail the CAN solution and the CAN products designed to support in-vehicle networking needs in the 80's and 90's. Further, the paper will discuss the CAN development tools and show network implementations and examples from a system designers perspective.

THE REASON

Based on the work performed by individuals participating in the ATA and SAE multiplex subcommittees [4,5], in-vehicle networking, is thought to provide a number of significant benefits to automotive manufacturers and customers, including: lower cost, higher quality, and improved product performance. Lower cost is obtained through reduced number of sensors, reduced wiring, and reduced manufacturing complexity. Higher quality is obtained through reduced number of components in the vehicle and increased diagnostic capability for manufacturing and for service. Improved product performance is obtained through system-level control strategy in the vehicle and better capability to present status and warning information to the driver interface.

The above benefits span the breath of in-vehicle networking applications, and are not equally weighted for all applications. The benefit obtained depends on the application and the implementation of the application. The SAE Multiplex Subcommittee has partitioned, for passenger cars, the applications into three major areas[6]: Class A - body electronics control; Class B - non-critical information sharing; and Class C - real time control. The ATA efforts also have performed partitioning by targeting the initial application range of Recommended Practices J1708 and J1587.

What is perhaps equally important to note, in addition to partitioning, is that the requirements from a network perspective are very close across the range. While each of these classes

(A,B,C) addresses different functions, and have been partitioned in that manner, they have many of the same needs, including: multi-master system; low latency; minimum CPU burden; prioritized message arbitration; and error detection.

It is likely that the applications will be grouped on a reduced number of buses (2 or possible 1), to achieve lower cost and reduced network complexity. The Controller Area Network (CAN) solution has been designed to meet the requirements of a broad range of applications within the automobile. The protocol has been implemented in a manner that allows use without being restricted to a single bus rate or vehicle application-type, as well as maintaining the transparency important to long-term designs.

THE PROTOCOL

The Controller Area Network (CAN) communication protocol is a high-performance contention-based serial communication solution which provides: open system flexibility, minimum CPU burden with capability to operate at up to 1 Mb/S, and error detection and failure diagnosis.

System Flexibility

In in-vehicle networking, "open system flexibility" is the ability to add or subtract modules transparently to and from the network. As an example, if a vehicle program added a module in its next generation which requires information already provided by existing modules in the network, those modules supplying the data should not need to be modified to share the parameter with a new node. "Receiver independence" is perceived as an important feature and has been integrated into the CAN message frame format (Figure 1).

The CAN message is composed of fields. The first field is the START-OF-FRAME, a single bit field which is responsible for synchronization and is important for arbitration. Following the START-OF-FRAME bit is the ARBITRATION-FIELD, an 11-bit field. It is dedicated to providing information about the contents of this individual message as well as message priority which is used in arbitration. The field has been designed to allow use in the manner that serves the system designers requirements best. Potential uses include: indicating a single parameter such as "engine speed";

indicating the origin of the parameter and the parameter itself such as "transmission module - vehicle speed" (similar to the ATA J1587 format); or indicating a grouping of parameters such as "door lock status (1,2,3,4)". The key attributes are: flexibility; the ability to arbitrate simultaneous message transmissions based on priority; receiver independence; and simplicity of design.

Message continuity and consistency are supported by the subsequent fields. The RTR-BIT allows Remote Transmission Requests from any node to the system. This provides the capability to request information in addition to the standard broadcast capability. The node requesting the information simply sends the correct ID in the ARBITRATION-FIELD and sets the RTR-BIT. The node on the bus responsible for that information will then broadcast it out onto the bus. This also allows powerful diagnosis capability by allowing nodes to detect or determine if their primary supplier for a specific parameter(s) is not functional. The following fields, CONTROL and DATA, provide information about the message and then the data itself. The CRC-FIELD (Cyclic Redundancy Check) provides strong error detection capability, and the ACK-FIELD allows the transmitter to be informed that the message has been received and that no errors were detected.

Multi-Mastership and Message Arbitration

The network must allow timely response to the dynamic and asynchronous actions of the vehicle as well as insuring predictable and minimum latency. For these reasons traditional master/slave and token-passing protocols have not been well received for automotive applications, and "random offset" contention-based protocols such as CSMA/CD (Carrier Sense, Multiple Access / Collision Detect) have negative factors in their arbitration mechanism. CAN is a hybrid of the traditional contention protocols in that it merges the attributes of the peer network with bus-time-effective, non-destructive bitwise arbitration. Non-destructive bitwise arbitration allows messages transmitted simultaneously to be arbitrated in a prioritized manner. The winner of the arbitration continues to transmit, unaffected, and the loser(s) of the arbitration re-transmit at the next "bus idle" state.

This arbitration technique, which is a significant part of the SAE MUX proposed recommended practice, is performed via a two-state media such as a two wire, differential bus over which messages are sent. The two states may be seen as dominant and recessive. A dominant bit will overwrite a recessive bit transmitted at the same time. Because each transmitting node also monitors the bus, it is able to detect when it has lost arbitration. If it loses arbitration it backs off, becomes a receiver for the present message, and lets the higher priority message continue (see Figure 2).

Keeping the CPU Free

In the vehicle system, some processors are small and lightly burdened, while others are higher performance and significantly loaded in throughput performance. Class B or C applications require all 3 of these systems (body electronics, powertrain, chassis) to be linked and provide the same interface to the bus. A software interface requires an inconsistent contribution from different modules, and gives a throughput impact to processors that cannot afford additional unnecessary functions.

Compatibility between interfaces on a network is important, and perhaps the most critical parameter in the network design. A software driven interface allows inconsistencies which must be resolved through extensive debugging, consuming engineering resources and product development time.

For a communication protocol or network implementation to offer long term design capability, it must be CPU or processor performance independent. In other words, the performance of the serial interface in bus speed, error detection, message storage, etc., should not be directly related to the CPU performance. A hardware solution provides a generic interface to the bus which does not significantly impact the performance of the CPU, which is consistent for all nodes in actions and timing, and which allows expansion to higher bus rates in future vehicle programs if required, i.e. a "clean" interface. Primarily for the above reasons, CAN was designed as a hardware solution. The CAN functional partitioning may be seen in Figure 3 as three major blocks.

The combined Interface Management Processor (IMP) and Bus Interface Unit

(BIU) perform all of the message framing, serialization, error detection, transmission and reception, etc. in hardware and do so transparently to the CPU. Messages are accessed by the IMP for either storage (on reception) or retrieval (for transmission) through the shared RAM. The shared RAM provides dual access for both the CPU and the IMP, while protecting against simultaneous access to the same message, and is seen as part of the CPU's memory map. As a result, independent of whether the IMP and shared RAM are on the same device as the CPU, the interface to the bus is transparent and actions are consistent for all nodes.

Keeping Messages Straight

To provide assurance that data received will be the same as that sent, and that it may be used with confidence, CAN implements powerful error detection mechanisms focusing on a cyclic Redundancy Check (CRC) word and both positive and negative acknowledgement.

It is important to separate error "protection" from "detection". Error "protection", i.e. not letting errors be induced on the bus, is largely determined by the system designer through the selection of bus media, bus location (routing), etc. Error "detection" is the ability to detect errors that have been induced on the bus and alert nodes that an error has been found. Of the two areas, error detection is the most critical because it allows the network to be aware of flawed messages and perform the steps required to obtain the correct message. An ideal system couples the best of both error protection and error correction.

The combined error detection mechanisms of CAN insure that the residual probability not to detect errors is well below 3×10^{-5} . It is important to note that this error detection measurement comes on top of the base capability that CAN detects: all global errors; up to 5 randomly distributed local bit errors; up to 15 consecutive local bit errors; and any odd number of local bit errors [7].

The key difference in CAN's error detection system over other communication directions is that CAN alerts all nodes in the network that an error has been detected at the time the error is first recognized. As a result, non-productive bus time is minimized and data quality is

insured. Nodes are provided this information through two complimentary acknowledgement schemes. The positive acknowledgement at the end of each message (seen as ACK-FIELD in Figure 1) is a single-bit field complimented by delimiters in which nodes receiving a correct message and planning to store it may respond simultaneously. The negative acknowledgement is a series of dominant bits which will be sent by any node on the bus at the point that it detects an error in the message. This negative acknowledgement, or "error frame", informs the transmitting node and all receiving nodes that an error has been detected. As a result all nodes are aware of the status of each message, data consistency is assured between nodes, and recovery from errors may be kept below 25 μ S (at 1 Mbit/S). Message correction is performed through automatic retransmission, and as a result of the acknowledgement schemes a significant link in the total fault diagnosis chain is attained [8].

THE PRODUCT

This section details the functional implementation of CAN in the product, the actions taken to send and receive messages to and from the bus, and the implementation of the device itself including the product family it is growing into.

Implementation and Communication

Functional partitioning for communication is shown in Figure 4. For the CPU, in any communication transfer on the network, minimal effort is required because it is functionally coupled to the bus interface through the shared RAM. As a result, for a transmission or reception, the actions required by the CPU are little more than those it would normally make even if it were not part of the network. Outside of its normal data storage and retrieval from RAM, the CPU is required only to check if access is allowed and toggle a dedicated control bit for the parameter to insure message consistency.

Shared RAM Array

The shared RAM array is seen as part of the CPU's memory map. It is configured at device power-up for the Communication Objects needed by that node, and contains several control bytes to: show node and system status, allow easy access to

updated Communication Objects, and other actions. The buffer memory layout, in its basic form, is shown in Figure 5. The shared RAM array is composed of dedicated control registers, communication objects, and an end mark signifying the end of RAM used for communication. Each individual communication object is composed of the identifier, control bits, and data. As a result, the number of communication objects which may be configured in the shared RAM array is purely dependent on the number of bytes in the data field since objects are placed contiguously.

The Silicon Cell and Production Device

The CAN implementation in silicon is designed as a "cell" to allow the same logic (i.e. exactly the same functionality) to be integrated in a broad family of devices. The first device is a stand-alone peripheral. The peripheral interfaces to any standard microcontroller/processor and was designed first for that reason - to give the system designer network capability with processor architecture independence. The peripheral contains, on-chip: the full CAN bus interface, the shared RAM array (64 bytes), a clock generator (which may be driven either by a crystal or by the processors clock), and a programmable clock divider to allow a broad range of bus speed options. The peripheral interfaces to the processor through the multiplexed address/data port, and as such the RAM array is seen as part of the CPU's memory map-maintaining the transparency critical to the automotive network and achieving high data throughput. To replace the I/O ports on the processor used by interfacing to the parallel bus, the ports are reconstructed on the peripheral. As a result there are 2 peripheral versions, the 40/44 (DIP/PLCC) lead version with port reconstruction and the 24 lead version without port reconstruction. Examples of the device with the 80C51 and the 6801 are shown to give an idea of the interface to a microcontroller (see Figures 6 and 7 respectively).

Future Products

The CAN logic cell may be readily integrated into control-oriented single chip microcontrollers. The integration provides the most cost effective implementation for CAN in networking long term. The resultant device will communicate using the CAN serial

870823

interface rather than a UART or SCI and thus achieve the full benefit of CAN with minimum die size and pin count. Functionally this device will look like Figure 4, with the added attribute that the user may now additionally use part of the existing RAM for communication if required.

Beyond Intel CPU-based architectures, it is anticipated that CAN will be integrated into microcontroller/processor devices supplied by other semiconductor manufacturers. This allows standardization in use and the ability to use a broad range of control architectures.

THE IMPLEMENTATION

This section shows the use of CAN in a range of applications. It is not within the scope of this paper to propose network partitioning (i.e. what nodes and/or messages go on what bus). Examples follow for status-type information sharing, real-time control, and body electronics control.

Real-Time Control, Non-Critical Parametric Sharing

Real-time control communications and non-critical parametric sharing have a key factor in common, parametric data. While the acceptable latency for the two classes may not be the same, by as much as an order of magnitude, much of the data passed is common to both: engine speed, vehicle speed, coolant temperature, braking status, etc.

The two "classes" will likely see evolutionary expansion in vehicle programs in terms of the number of nodes and overall bus load they must support, hence a long-term solution is required which allows the manufacturer to integrate a network into the vehicle which will meet the system needs for several years. For simplicity the example has been shown in one common network in Figure 8.

Body Wiring Control

For body wiring electronics control of functions such as lights, etc. a full CPU-based microcontroller is not always required and in some cases too costly. Using the stand-alone peripheral (the 82C751) already developed, a cost conscious Serially-Linked I/O (SLIO)

network may be implemented without using the traditional method of placing additional burden on the body computer. A system configuration which might serve as an example is shown in Figure 9.

Developing the Network

Developing an in-vehicle network is not an easy task. It requires integrating communication between several modules, insuring that the correct messages arrive at the correct nodes, that latency requirements are met, etc. To support network design, CAN has 2 development tools: a software simulator and a CAN hardware emulator.

The software simulator is primarily a tool for the system designer. It is a user-friendly package which uses inputs such as: number of nodes; bus speed; bus length; noise signals; number of messages; etc. In return, the simulator gives a detailed statistical output of the system performance including: latency for the system, each node, each message; bus load; number of messages affected by noise; number of arbitrations and outcome for each node, each message; etc. This gives the system designer the opportunity to adjust the network to the desired performance before it ever gets in the vehicle.

The hardware emulator does what the simulator cannot - test hardware on the bench, in the vehicle, and begin prototyping. The single-board solution emulates the CAN stand-alone peripheral through discreet logical devices, providing a development module now. The board implements the full CAN protocol using a combination of microcontrollers and Electrically Programmable Logic Devices (EPLDs) and as a result will communicate up to 50K bits/second.

The Solution

CAN has been designed for and is dedicated to in-vehicle networking. It has been designed to meet the networking requirements of: lower cost, higher quality, and improved product performance. Further it provides a high performance solution, having a programmable bit rate up to 1 Mb/s. These attributes are achieved through the hybrid combination of traditional contention protocols and non-destructive bitwise arbitration, and by placing the protocol in hardware to achieve transparency and the capability for high performance. A high level of error detection is also provided. Further, development tools are provided which allow system analysis and modification in both software and hardware to achieve efficient program development.

One of the most important features of the protocol is its readiness. At the time of this paper, the CAN peripheral will have been sampled and be nearing production capability. The result is a solution for tomorrow, present today.

The hardware emulator does what the simulator cannot - test hardware on the bench, in the vehicle, and begin prototyping. The single-board solution emulates the CAN standard peripheral through direct logical devices, providing a development module now. The board implements the full CAN protocol using a combination of microprocessors and electrically programmable logic devices (EPLDs) and as a result will communicate up to 30K bauds/second.

REFERENCES

- [1] ATA-TMC/SAE Truck & Bus Data Format Subcommittee, J1587 Draft, Recommended Practice for Electronic Data Interchange Between Microcomputer Systems in Heavy Duty Vehicle Applications, June, 1986.
- [2] F. Miesterfeld, Chrysler Corporation, "Chrysler Collision Detection (CCD) - A Revolutionary Vehicle Network," February, 1986, SAE International Congress and Exposition.
- [3] W. Johnson and J. Volk, Ford Motor Company, "A Proposal for a Vehicle Network Protocol Standard," February, 1986, SAE International Congress and Exposition.
- [4] SAE Multiplex Subcommittee, Proposed Recommended Practice, "Proposed Class 'B' Bus Interface," January, 1987.
- [5] ATA-TMC/SAE Truck & Bus Electronics Interface Subcommittee, J1708 Draft.
- [6] SAE Multiplex Subcommittee Automotive Task Force, "Chart of Vehicle Multiplexing Characteristics," February, 1986.
- [7] U. Kiencke, S. Dais, and M. Litschel, "Automotive Serial Controller Area Network," February, 1986, SAE International Congress and Exposition.
- [8] Intel/Bosch Controller Area Network C-Specification, January, 1986.

270509-7

The two "classes" will likely see evolutionary expansion in vehicle programs in terms of the number of nodes and overall bus load they must support, hence a long-term solution is required which allows the manufacturer to integrate a network into the vehicle which will meet the system needs for several years. For simplicity the example has been shown in one common network in Figure 8.

Body Wiring Control

For body wiring electronics control of functions such as lights, etc. a full CPU-based microcontroller is not always required and in some cases too costly. Using the stand-alone peripheral (the 82C751) already developed, a cost conscious serially-linked I/O (SLIO)

870823

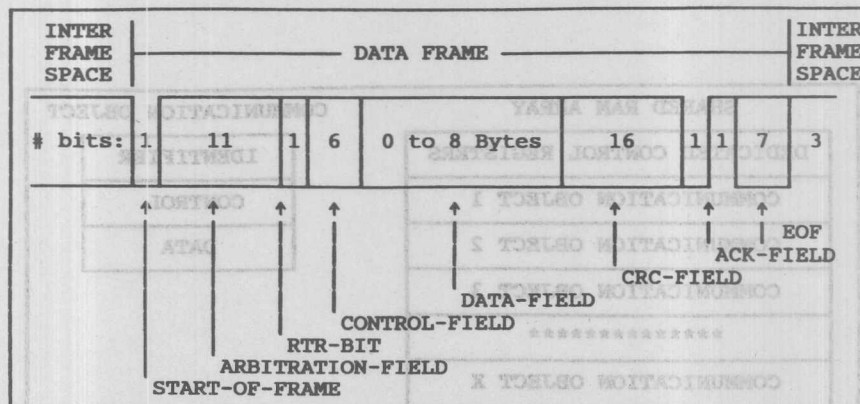


FIGURE 1: CAN MESSAGE FRAME FORMAT

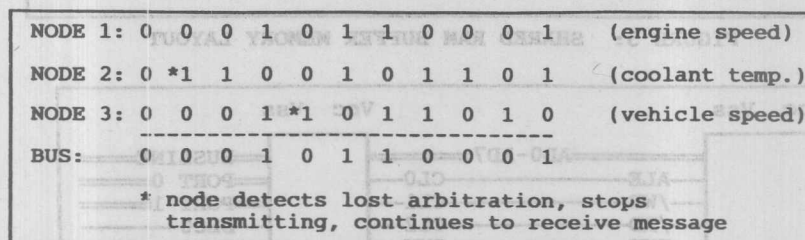


FIGURE 2: NON-DESTRUCTIVE BITWISE ARBITRATION

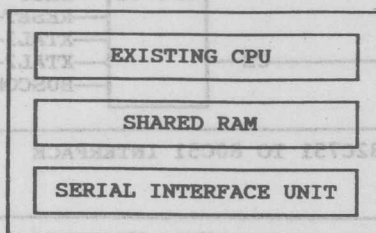


FIGURE 3: CAN FUNCTIONAL PARTITIONING

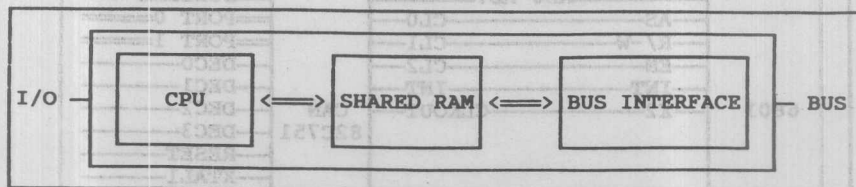


FIGURE 4: CAN FUNCTIONAL IMPLEMENTATION

270509-8

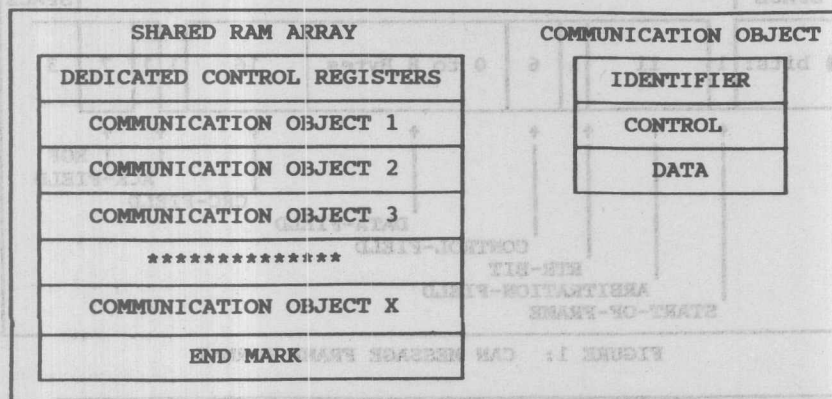


FIGURE 5: SHARED RAM BUFFER MEMORY LAYOUT

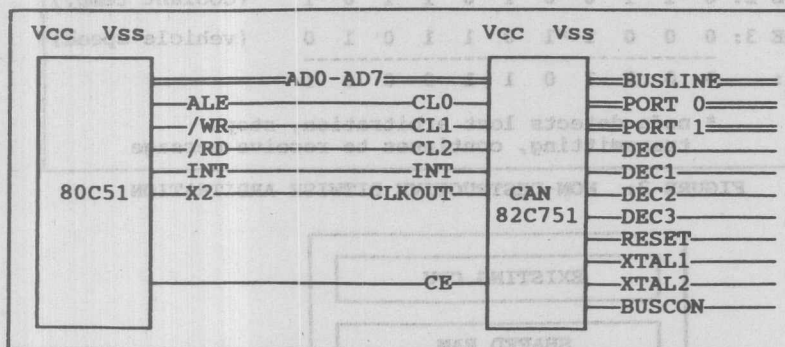


FIGURE 6: 82C751 TO 80C51 INTERFACE

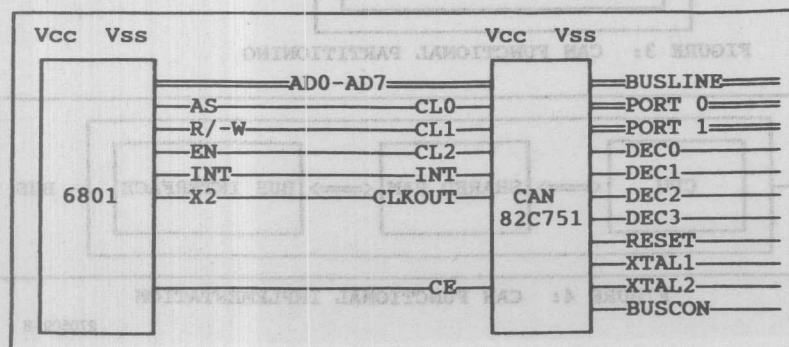


FIGURE 7: 82C751 TO 6801 INTERFACE

870823

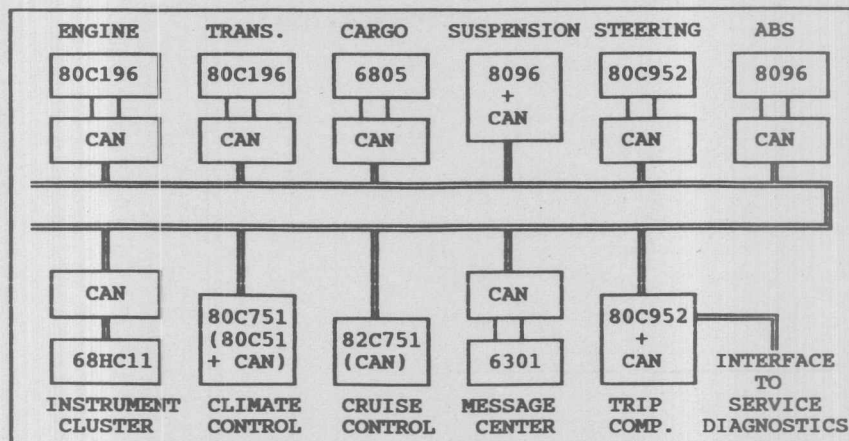


FIGURE 8: REAL TIME CONTROL AND NON-CRITICAL PARAMETRIC SHARING

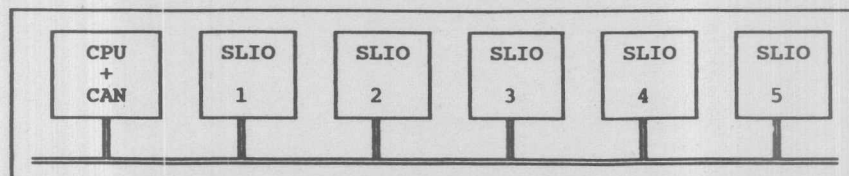


FIGURE 9: SERIALY LINKED I/O (SLIO) MODE NETWORK EXAMPLE

270509-10



82526 CONTROLLER AREA NETWORK CHIP ARCHITECTURAL OVERVIEW AUTOMOTIVE

Interface Management Processor (IMP), Bit Stream Processor (BSP), Bus Timing Logic (BTL), Transceiver Logic (TCL), Processor Interface Unit (PIU), Error Management Logic (EMI), Clock Generator and a dual port RAM. These hardware modules implement all necessary features of a high performance serial communication protocol. When connected to a microprocessor and simple line driver, the 82526 performs the principal functions of the physical and data link layer. Figure 1 shows a block diagram of the 82526.

82526 / Host CPU Interaction

The 82526 uses an 8-bit multiplexed address and data bus optimized for operating with Intel's microcomputers and microprocessors. Other architectures may be used due to the small number of handshake signals required by the 82526.

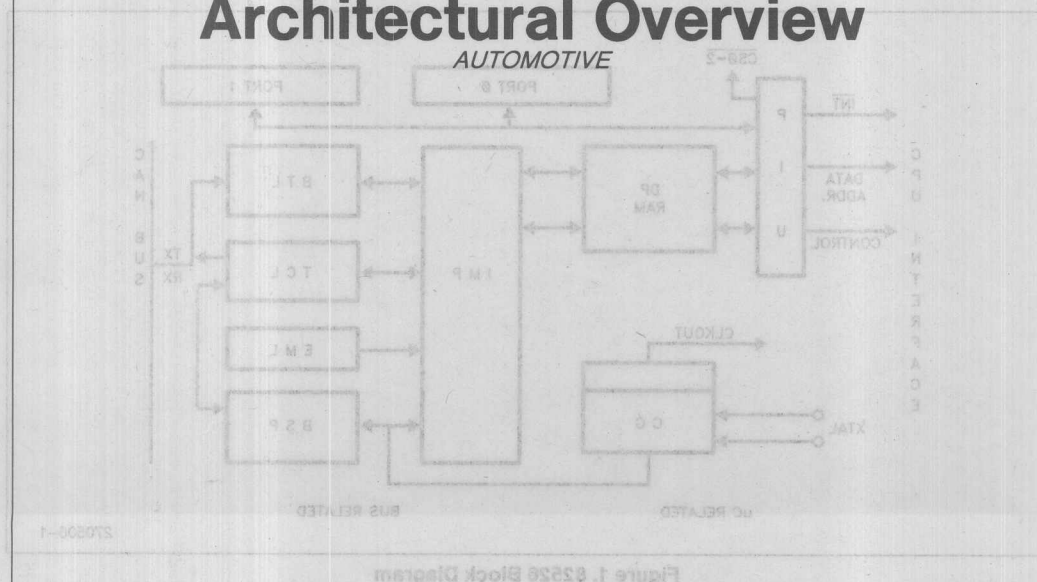
As shown in Figure 1, the BSP, TCL, BTL, and EMI are related with Bus Line Logic. The IMP, RAM, and PIU are related to the CPU Interface Logic. The logic blocks BSP, BTL, TCL, and EMI are referred to as the "Serial Interface Unit".

The 82526 communicates with the CPU through the on-chip memory (RAM) which includes a global control register. The control register is used to manage the global control and the CPU interface logic.

82526

Controller Area Network Chip Architectural Overview

AUTOMOTIVE



1.0 GENERAL FEATURES

- Multimaster Architecture
- Bus Access Priority by Message
- 2032 Different Messages
- Guaranteed Latency Time for High Priority Messages
- Powerful Error Handling
- Data Length up to 8 Bytes
- Configuration Flexibility
- Non-Destructive Bitwise Arbitration
- Two 8-Bit Ports
- NRZ Coding/Decoding with Bit Stuffing
- Programmable Transfer Rate up to 1Mbit/sec
- Programmable Output Driver Configuration
- Programmable Clock Output
- 44 Pin PQCC
- Additional CS Outputs

February 1988

82526

CONTROLLER AREA NETWORK CHIP

ARCHITECTURAL OVERVIEW

AUTOMOTIVE

1.0 GENERAL FEATURES

- Multimaster Architecture
- Bus Access Priority by Message
- 2032 Different Messages
- Guaranteed Latency Time for High Priority Messages
- Powerful Error Handling
- Data Length up to 8 Bytes
- Configuration Flexibility
- Broadcast Message Transfer
- Non-Destructive Bitwise Arbitration
- Two 8-Bit Ports
- NRZ Coding/Decoding with Bit Stuffing
- Programmable Transfer Rate up to 1MBit/Sec
- Programmable Output Driver Configuration
- Programmable Clock Output
- 44 Pin PLCC
- Additional CS Outputs

1.1 FUNCTIONAL OVERVIEW

The 82526 Communication Controller is a highly integrated VLSI device, based on automotive in-vehicle networking requirements. Included on the chip are an

Interface Management Processor (IMP), Bit Stream Processor (BSP), Bus Timing Logic (BTL), Transceiver Logic (TCL), Processor Interface Unit (PIU), Error Management Logic (EML), Clock Generator and a quasi dual Port RAM. These hardware modules implement all necessary features of a high performance serial communication protocol. When connected to a microprocessor and simple line drivers, the 82526 performs the principal functions of the physical and data link layer. Figure 1 shows a block diagram of the 82526.

82526 / Host CPU Interaction

The 82526 uses an 8-bit multiplexed address and data bus optimized for operating with Intel's microcontrollers and microprocessors. Other architectures may be used due to the small number of handshake signals required by the 82526.

As shown in Figure 1, the BSP, TCL, BTL, and EML are related with Bus Line Logic. The IMP, RAM, and PIU are related to the CPU Interface Logic. The logic blocks BSP, BTL, TCL, and EML are referred to as the "Serial Interface Unit".

The CPU communicates with the 82526 through the 82526 on-chip memory (RAM) which includes a global status and control register. The on-chip memory serves as the communication buffer interface between the CPU and the IMP. The CPU initializes the global

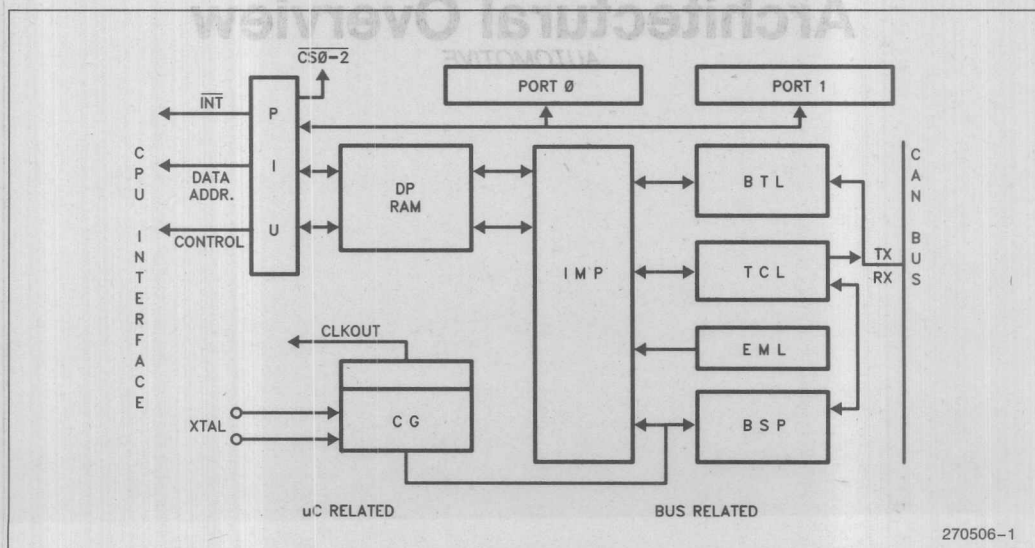


Figure 1. 82526 Block Diagram

Pin	Description
CS	Chip select input pin. When this signal is low, the 82526 is selected by the CPU for transfer of command, status or data to or from the internal RAM, data to or from Port 0 and Port 1, or access to additional external devices within the address space 64-255 (see CS0 to CS2). The direction of the data flow is determined by the RD or WR inputs.
INT	Interrupt output pin. An active low signal indicates to the CPU that the 82526 is requesting an interrupt. The interrupt output is an OR-function between the EXTINT and or internal generated interrupt signals.
RX0, RX1	Serial data bus input pins to the on-chip input-comparator.
TX0	Serial data bus output pin from the on-chip programmable output-driver 0.
TX1	Serial data bus output pin from the on-chip programmable output-driver 1.
RDY	Ready output pin used to control access to the internal RAM either by a CPU or by the Interface Management Processor (IMP). Ready output is active low. A pull up resistor keeps this pin high during reset.
ALE	Address latch enable input.
PORT 0	8-bit quasi-bidirectional I/O port (TTL/LS compatible).
PORT 1	8-bit quasi-bidirectional I/O port (TTL/LS compatible).
AD0-AD7	Multiplexed Address/Data bus. AD0-AD7 are bidirection three state lines connected to the system's Data/Address Bus for transfer of data, commands and status.
CS0	Address decoding. Chip select 0 output pin is activated low if CS is set low and an address between 40H-BFH is latched into the 82526 by a CPU. CS0 remains low for the time ALE is active low.
CS1	Address decoding. Chip select 1 output pin is activated low if CS is set low and an address between C0H-DFH is latched into the 82526 by a CPU. CS1 remains low for the time ALE is active low.

Pin	Description
CS2	Address decoding. Chip select 2 output pin is activated low if CS is set low and an address between E0H-EFH is latched onto the 82526 by a CPU. CS2 remains low for the time ALE is active low.
VSS1	Digital GND.
VSS2	Analog GND.

NOTE:

CS0, CS1 and CS2 are push-pull outputs, and may be used to select additional system peripherals.

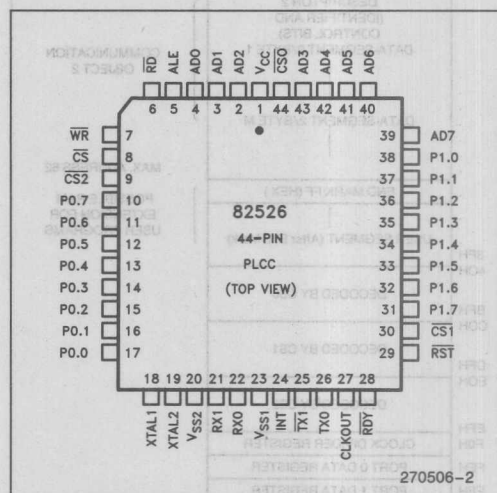


Figure 2. 44-PIN PLCC Package

3.0 FUNCTIONAL DESCRIPTION

3.1 BUFFER MEMORY LAYOUT

The 82526 on-chip RAM (communication buffer) is seen as part of the CPU's memory map and may be defined as a "decoupling device" between the CPU and the "Bus-Interface" of the 82526. The buffer memory area used for communication is configured by the user during an initialization download after power-up. It consists of dedicated control registers, communication objects, and an end mark signifying the end of RAM area used for communication. Each individual communication object is composed of the identifier, control bits, and data. The CPU programmer operates only on this communication buffer to perform message transfers, while the "Bus Interface Logic" of the 82526 manages the bus traffic. The buffer memory layout is shown in Figure 3.

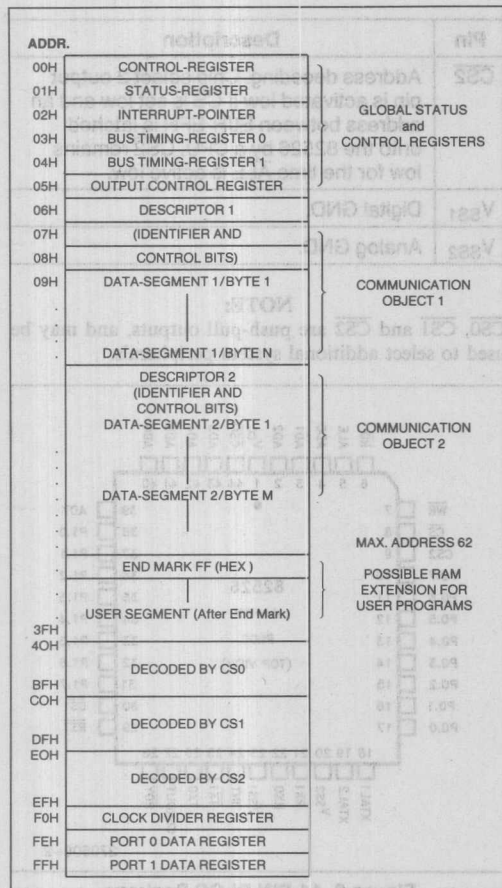
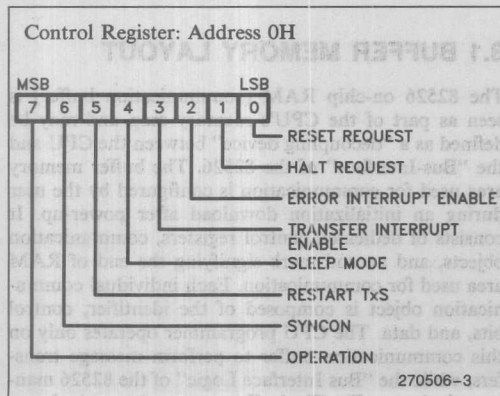


Figure 3. Buffer Memory Layout

3.2 CONTROL REGISTER



Reset Request

RESET REQUEST is set by the CPU and read by the IMP. If set to "high" and detected by the IMP, it causes an SIU (Serial Interface Unit) internal hardware reset. The execution of the reset request is acknowledged to the CPU via the reset status bit (see Status Register), set by the IMP indicating to the host that the 82526 is ready for initialization. Thereafter, the IMP remains in a "loop mode" waiting for the reset request to be set "low" by the host after completion of the RAM configuration. The 82526 returns to the operating mode immediately after the reset status bit is set "low" by the IMP. During the reset sequence, the PIU, clock generator and the buffer memory remain active.

NOTE:

For proper operation, the reset request bit must be set to "high" for any start-up or new RAM configuration routine and set to "low" after the End Mark.

Reset request set "high" does not break a transmission or reception of a message in process, but will stop the 82526 from transmitting or receiving the next message.

Halt Request

HALT REQUEST is set by the CPU and read by the IMP. When set to "high" the 82526 continues processing the current transmission or reception and then stops any further activity until the halt request bit is set to "low" which restarts normal operation. Halt request forces the IMP to stay in the idle-loop while the PIU, SIU and clock generator remain active.

NOTE:

In order to change the buffer memory configuration, use reset request rather than halt request to stop the IMP. It must be noted, however, that reset request will also result in a reset of the EML logic and, therefore, will cancel the current EML error history.

Error Interrupt Enable

ERROR INTERRUPT ENABLE is set by the CPU and read by the IMP. If set to "high", the interrupt output signal of the 82526 to the CPU is enabled, and disabled if set to "low". The 82526 may generate an error interrupt for the following reasons:

- error status bit set (Status Register)
- bus status set to “off bus” (Status Register)
- ram status set to “error” (Status Register)

Transfer Interrupt Enable

TRANSFER INTERRUPT ENABLE is set by the CPU and read by the IMP. If set to "high", the 82526 will generate an interrupt to the CPU after a message is successfully received or transmitted and the transfer interrupt enable bit within the descriptor of the corresponding communication object is set. If set to "low", any transfer interrupt to the CPU is disabled independent of the transfer interrupt enable bit within the descriptor.

Sleep Mode

SLEEP MODE is set by the CPU and read by the IMP. If the bit is set to "low" (normal operation) the 82526 will not enter the sleep mode. If set to "high", the device may enter into the sleep mode based on bus traffic (both, transmission and reception). A return from the sleep mode to the normal operation mode may be activated by the following events:

- start bit detection
- any CPU access to the 82526 on-chip RAM, Port 0 or Port 1, or to the clock divider register

An attached host CPU does not see the 82526 in the sleep mode. Hence, all timings for RAM and I/O access remain valid for the sleep mode.

The 82526 enters the sleep mode when 256 recessive bit-times (bus idle) have elapsed and enable signals from all functional blocks are present. This allows sufficient time for the IMP to terminate current processes.

Restart TxS

(Restart transmit search.) Any write access by the CPU to set this bit to "high" forces the IMP to start a new transmit search loop at the first Communication Object. It is not necessary to reset this bit between "restart transmit search" requests.

Syncom

SYNCOM is set by the CPU and read by the IMP. This bit controls resynchronization of the Bit Timing Logic during the reception of a frame. If set to "high", bus line transitions from "recessive" to "dominant" as well as from "dominant" to "recessive" are used for resynchronization.

If set to "low", only transitions from "recessive" to "dominant" are used for resynchronization.

NOTE:

It is not recommended to resynchronize on both transitions at very high speed baud rates.

Operation

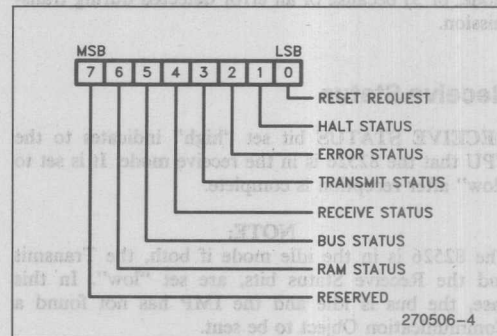
OPERATION bit must be set to "low" by the CPU during initialization. It will force the 82526 to start operation as soon as the reset status bit is reset by the IMP. This bit must never be set to "high" by a CPU because it will result in an unpredictable mode.

3.3 STATUS REGISTER (Address 01H)

The Status Register is modified by the IMP only. After power up reset and/or reset request from the CPU (see reset request), the IMP will set the reset status bit "high". It is set "low" after the reset request is released by the CPU. A CPU write is not allowed to this register. A read access to this register will report the reset status of the 82526.

NOTE:

After start-up, bits 1 to 7 of the status register are random until reset status (bit 0) is set by the IMP.



Reset Status

RESET STATUS bit set to "high" by the IMP acknowledges to the CPU that the reset request from the CPU was detected and an internal hardware reset is being performed by the 82526. The bit is set back to "low" by the IMP after the CPU has reset the reset request bit to "low" and forces the 82526 to start or restart normal operation.

Halt Status

HALT STATUS bit is altered corresponding to the Halt Request from the CPU. It is set "high" by the IMP to indicate the Halt Status (idle Loop) was entered, and set to "low" when the Halt Request is released by the CPU. If Halt Request is released when the Serial Interface Unit (SIU) is in the reception mode, the IMP remains in a wait loop until the next interframe space time.

Error Status

ERROR STATUS bit set to "high" by the IMP indicates to the CPU that the error management logic (EML) has detected a major problem either with transmission or reception of messages. If more than one message out of eight messages as an average are consecutively directed at this node, a local error may occur. The EML observes the occurrence of receive and transmit errors and takes the appropriate actions (see error confinement for more details). The Error Status bit is normally set "low".

Transmit Status

TRANSMIT STATUS set "high" indicates to the CPU that the 82526 currently is in the transmit mode. A "low" is set when 1) the 82526 enters the idle mode after a successful transmission, 2) there was a loss of the arbitration during transmission of the message identifier forcing the 82526 to immediately enter the receive mode, or 3) because of an error detected during transmission.

Receive Status

RECEIVE STATUS bit set "high" indicates to the CPU that the 82526 is in the receive mode. It is set to "low" after reception is complete.

NOTE:

The 82526 is in the idle mode if both, the Transmit and the Receive Status bits, are set "low". In this case, the bus is idle and the IMP has not found a Communication Object to be sent.

Bus Status

BUS STATUS is modified by the IMP and read by the CPU. If set to "low", the 82526 is bus active (on - bus). If set to "high" (off - bus), the 82526 remains in the bus-off mode until a reset request is set by the CPU and executed by the IMP (see reset request bit and error management for more details).

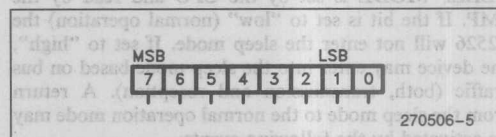
RAM Status

RAM STATUS bit normally is set to "low" by the IMP after a complete RAM configuration set up by the CPU and no buffer memory configuration inconsistency is detected by the IMP. The bit will be set to "high" if a buffer-memory inconsistency (faulty communication object) is detected after the 82526 has started operation.

3.4 INTERRUPT POINTER (Address 02H)

The Interrupt Pointer is an 8-bit register that contains either the address of the Status Register (error related interrupt), or the RAM address of a communication object with highest priority for which the conditions Transfer Interrupt Enable is enabled ("high") and Transfer Status is completed ("high").

The CPU acknowledges an 82526 interrupt by writing an empty pointer (FFH) to this register. If the CPU itself is polling an empty pointer (read access) from the Interrupt Pointer register, this value is interpreted as if no interrupt has occurred.

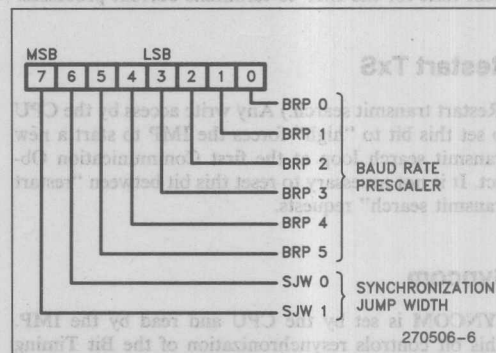


NOTE:

The interrupt pointer must be set to "FFH" by the CPU during initialization. If an external interrupt occurs, this register remains unchanged (FFH).

3.5 BUS TIMING REGISTER 0 (Address 03H)

The baud rate prescaler and synchronization jump width are programmed by the Bus Timing Register 0.



Synchronization Jump Width

These two bits are introduced to compensate for phase shifts between clock oscillators of different bus nodes. Any node may resynchronize to a relevant transition of the bus bitstream of a transmitter.

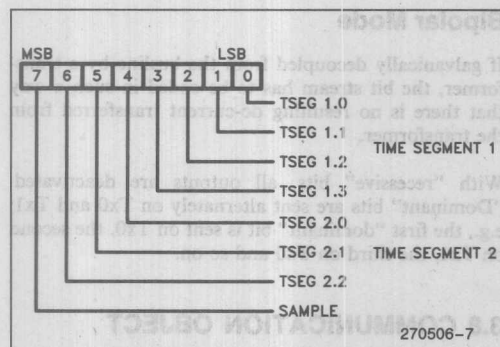
The Synchronization Jump Width defines the maximum number of BTL cycles. A bit may be shortened or lengthened by one resynchronization during transmission of a Data Frame or Remote Frame.

Table 1. Baud Rate Prescaler

Baud Rate Prescaler BRP 0 - BRP 5	BTL Cycle Time
000000	1 × System Cycle Time
000001	2 × System Cycle Time
000010	3 × System Cycle Time
⋮	⋮
111111	64 × System Cycle Time

3.6 BUS TIMING REGISTER 1 (Address 04H)

The number of samples per bit, the Delay Time and the Phase Shift Buffer are programmed by Bus Timing Register 1.



BUS SAMPLE: This bit determines whether a bit is sampled directly by the BTL or if each bit first is filtered by the majority logic. If SAMPLE is set to "low", a bit is sampled once by the BTL, and if set to "high", three samples per bit are taken.

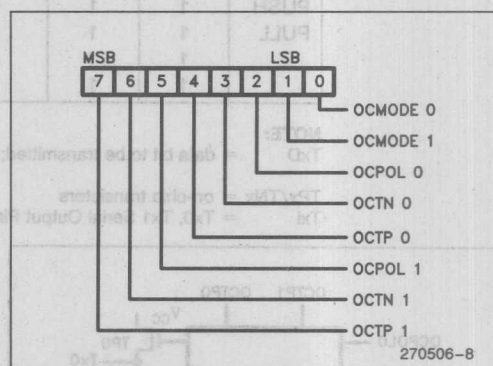
NOTE:

The majority logic (SAMPLE = "1") has the same effect as a delay of one BTL cycle on the bus line. This has to be taken into account for the calculation of time segment 1 and 2 as well as for synchronization jump width. (For more information see Bit Timing Logic.)

TIME SEGMENT 1 / TIME SEGMENT 2: Time Segments within a bit time are introduced to determine the number of BTL cycles per bit time and the location of the sample point within a bit time. Also see minimum configuration of the BTL.

3.7 OUTPUT CONTROL REGISTER (Address 05H)

The Output Control Register allows set up of different output driver configurations under software control. The user may select active pull-up, pull-down, float, or push-pull output by reviewing Tables 2 and 3.



Programmable Features

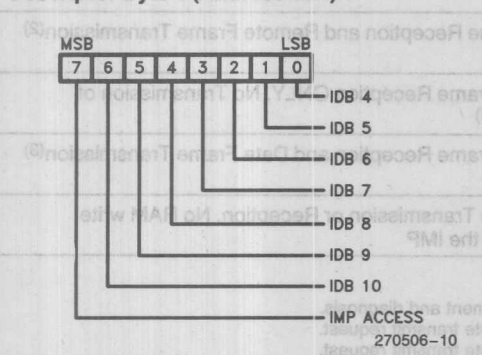
Table 2. Programmable Output Functions

OCMODE 1	OCMODE 0	Function
1	0	Normal Mode Bit stream transmitted on both, Tx0 and Tx1
1	1	Normal Mode Tx0: Bit Stream Tx1: Bus Clock (bclk)
0	0	Bipolar Mode
0	1	Not Used

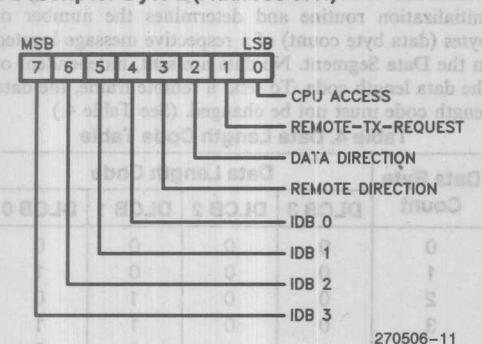
Descriptor Bytes 1, 2 and 3

The Descriptor consists of three bytes, assigned to each Communication Object by the user. The three bytes include the message identifier and a control segment which contains semaphore bits to guarantee mutual exclusion of access to the Communication Buffer, if required.

Descriptor Byte 1 (Address 06H)



Descriptor Byte 2 (Address 07H)



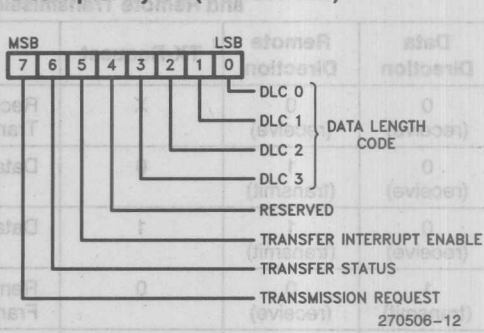
NOTE:

A CPU write access to Descriptor Byte 2 should be executed only if the reset request bit was set by the CPU and acknowledged by the IMP via the halt status bit.

Identifier

An Identifier is a unique name for each Communication Object and defines the corresponding, specific message. Its digital number also determines the priority of each single message for bus access (see arbitration).

Descriptor Byte 3 (Address 08H)



The Identifier consists of eleven bits (IDB 0 to IDB 10). IDB 10 is the most significant bit, transmitted first following the start bit. The priority is defined to be highest for the lowest binary number.

NOTE:

IDB 10 to IDB 4 set to "high" can not be used in an Identifier because this code serves as end-mark. This results in a total number of 2032 different Identifiers (messages), starting from 00000H (highest priority) to 07EFH (lowest priority).

Control Bits

IMP ACCESS is set by the CPU and read by the IMP. If set to "low", IMP access to the data of this Communication Object is "locked". If reset to "high", IMP access is released.

CPU ACCESS is set by the IMP and read by the CPU. If set to "low", CPU access to the data of this Communication Object is "locked". If reset to "high" CPU access is released. In case of a conflict, the priority is assigned to the CPU access.

REMOTE TX REQUEST is modified by the IMP only! If a remote frame has been received, the appropriate Data Frame will be transmitted if 1) the data direction bit is set "high" (transmit), and 2) the remote direction is set "low" (receive). The Remote Tx Request bit represents an internal flag used by the IMP to process the reception of remote frames. This bit must not be modified by a host, except during initialization (set to "low").

DATA DIRECTION is set by the CPU and read by the IMP. If set to "low", a Communication Object is assigned to receive a message of the respective Identifier. If set to "high", a Communication Object is assigned for transmission. The appropriate message will be transmitted if Transmission Request is set "high" and Remote Direction is set "low". (See Table 5.)

Table 5. Data Direction / Remote Frame Direction / Transmission Request and Remote Transmission Request Bit Combinations.

Data Direction	Remote Direction	TX-Request	Function
0 (receive)	0 (receive)	X	Reception of Data Frame or Remote Frame but no Frame Transmission ⁽¹⁾
0 (receive)	1 (transmit)	0	Data Frame Reception ONLY. No Frame Transmission ⁽²⁾
0 (receive)	1 (transmit)	1	Data Frame Reception and Remote Frame Transmission ⁽²⁾
1 (transmit)	0 (receive)	0	Remote Frame Reception ONLY. No Transmission of Frame's. ⁽³⁾
1 (transmit)	0 (receive)	1	Remote Frame Reception and Data Frame Transmission ⁽³⁾
1 (transmit)	1 (transmit)	X	NO Frame Transmission or Reception. No RAM write access by the IMP

NOTES:

1. Useful in order to monitor bus activities during system development and diagnosis.
2. Normal Operation for reception of data or transmission of remote transmit request.
3. Normal Operation for transmission of data or reception of remote transmit request.

REMOTE DIRECTION set by the CPU and read by the IMP. If set to "high" and Data Direction is set to "low", a CPU may initiate the transmission of a Remote Frame when Transmission Request is set "high". (See Table 5.)

TRANSMISSION REQUEST is modified by the CPU and IMP. If set to "high" by the CPU, a Data Frame or Remote Frame will be transmitted based on which bit (Data Direction or Remote Direction) is set to "transmit". Transmission Request is reset by the IMP after a successful transmission.

TRANSFER STATUS is modified by the IMP and the CPU. This bit is set "high" (82526 transfer complete) by the IMP after a successful transmission or reception. The Transfer Status Bit is reset by the CPU in order to signal to the IMP that the CPU transfer is complete.

NOTE:

If both, Data Direction and Remote Direction are set "high", the corresponding data segment of a communication object is locked for IMP access.

TRANSFER INTERRUPT ENABLE is modified by the CPU. If set "high" (enable), the interrupt signal will be activated if the Transfer Status of a corresponding Communication Object was found complete by the IMP. No interrupt is generated if this bit is set to "low" (interrupt disabled).

DATA LENGTH CODE is set by the CPU during the initialization routine and determines the number of bytes (data byte count) of a respective message located in the Data Segment. No data are sent, independent of the data length code. To send a remote frame, the data length code must not be changed. (See Table 4.)

Table 4. Data Length Code Table

Data Byte Count	Data Length Code			
	DLCB 3	DLCB 2	DLCB 1	DLCB 0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0

NOTE:

Data byte count above eight is reserved and should NOT be used.

DATA SEGMENT holds the data of the corresponding Communication Objective. The number of data bytes per segment is defined by the data length code in each descriptor.

ENDMARK set to FFH by the CPU indicates to the IMP that there is no Communication Object stored beyond the Endmark (max. address 62). The Endmark

indicates to the IMP the end of the linked list of Communication Objects. A Communication Object Identifier can not have 'FF'H for the most significant bits as it would conflict with the Endmark.

USER SEGMENT may follow the Endmark since it is not used by the 82526 controller. It is available to the user as general data memory and does not affect the function of the 82526.

4.0 82526 FRAME TYPES

The 82526 communication controller supports four different frame types:

- data frame
- remote frame
- error frame
- overload frame

There are two logical bit representations defined—"dominant" and "recessive".

A recessive bit on the bus line appears only if all connected nodes at that time send a recessive bit. If one or more nodes send a dominant bit, the bus line reflects a dominant bit.

A recessive bit is a high-level signal of a npn-open-collector bus.

A dominant bit always overwrites a recessive bit if emitted on the bus at the same time by a different node. If the electrical bus is implemented as an npn-open-collector bus, a dominant bit corresponds to a low level signal.

4.1 DATA FRAME

A Data Frame is composed of seven different fields:

- start bit
- arbitration field
- control (identifier) field
- data field (data segment)
- CRC field
- acknowledge field
- end of frame

START OF FRAME signals the start of a data- or remote frame. It consists of a single dominant bit used for synchronization of receiving nodes. The 82526 will start a transmission only if a bus-idle state is detected.

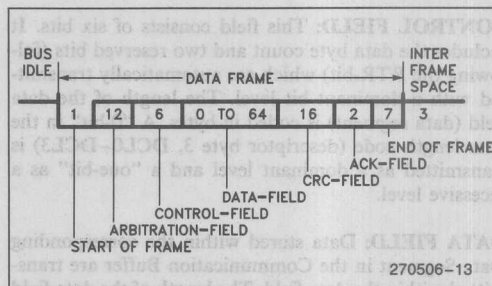


Figure 5. Data Frame Format

ARBITRATION FIELD consists of the message Identifier and one additional control bit (Remote Transmission Request).

The simultaneous message transmission start of two or more nodes is solved by bitwise arbitration during the transmission of the Arbitration Field (for more details see bitwise arbitration).

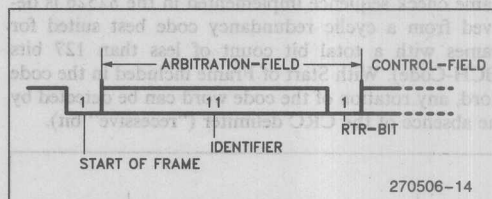


Figure 6. Arbitration Field Format

IDENTIFIER: This 11-bit field provides information about each individual message as well priority for the message. The Identifier defines the corresponding specific message (e.g., engine speed, temperature, pressure, etc.) of a Communication Object. Its digital value represents the message priority for bus access.

NOTE:

An Identifier does not define a physical station address to determine a Receiver of a frame on a multi-drop line. The 82526 controller decides, based on the acceptance filter process of a received Identifier, whether data being received within a correct frame are to be accepted or not.

Within an 82526 based communication network, there exists no discrimination between a point-to-point, multicast or broadcast communication.

RTR BIT: A station, active as a receiver may initiate the transmission of the data by transmission of a Remote Frame to the network, addressing the data source via the Identifier. Within a data frame, the RTR-bit is transmitted as a dominant bit level (for more information see Remote Frame).

CONTROL FIELD: This field consists of six bits. It includes the data byte count and two reserved bits (following the RTR-bit) which are automatically transmitted with a dominant bit level. The length of the data field (data segment) is coded in bytes. A "0-bit" in the data length code (descriptor byte 3, DCL0-DCL3) is transmitted as a dominant level and a "one-bit" as a recessive level.

DATA FIELD: Data stored within the corresponding Data Segment in the Communication Buffer are transmitted within the data field. The length of the data field varies in the range from 0 to 8 bytes based on the value of the data byte count (data length code). The byte at the lowest address of the data segment will be transmitted MSB first.

CRC FIELD contains the CRC check sum (15 bits) followed by the CRC delimiter (1 bit). The cyclic redundancy code includes the start bit (start of frame), arbitration field, control field, data field and CRC field. The most significant bit (MSB) is transmitted first. The frame check sequence implemented in the 82526 is derived from a cyclic redundancy code best suited for frames with a total bit count of less than 127 bits (BCH-Code). With Start of Frame included in the code word, any rotation of the code word can be detected by the absence of the CRC delimiter ("recessive" bit).

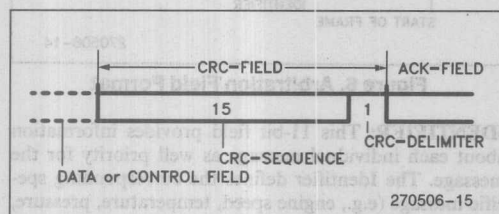


Figure 7. CRC Field Format

ACKNOWLEDGE FIELD: The ACK-FIELD consists of two bits, the ACK-SLOT and ACK-DELIMITER which are sent with a "recessive" level by the transmitter. Any receiving 82526 acknowledges to the transmitting 82526 a match of the complete/correct CRC check sum within the ACK-SLOT by superscribing the recessive bit with a dominant bit. A transmitter monitoring the bus level recognizes that at least one receiver within the system has received a complete and correct message (no error was found).

The ACK-DELIMITER (recessive bit) is the second bit of the ACK-FIELD. As a result, the ACK-SLOT is surrounded by two recessive bits—the CRC delimiter and the ACK delimiter.

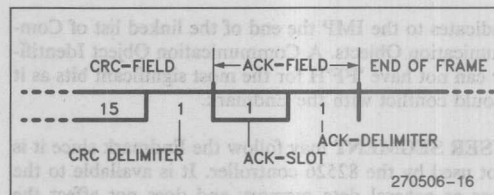


Figure 8. Acknowledge Field Format

END OF FRAME: Each Data Frame or Remote Frame is delimited by the end of frame bit sequence which consists of seven "recessive" bits (exceeding the bit stuff width by two bits). A receiver detects the end of a data frame independent of a previous transmission error because the receiver expects all bits up to the end of CRC-FIELD to be coded by the bit stuffing method. Bit stuffing is not used in end of frame.

4.2 REMOTE FRAME

A Remote Frame is composed of six different fields:

- start of frame
- arbitration field
- control field
- CRC field
- ACK field
- end of frame

Contrary to the Data Frame, the RTR-bit of the Remote Frame is "recessive" and no data segment is being transmitted independently of the Data Length Code set by the Descriptor of the corresponding Communication Object.

The RTR-bit allows Remote Transmission Requests from any node to the system. This provides the capability to request information in addition to the standard broadcast characteristics. It also supports powerful diagnostic capability by being able to determine if the primary supplier (data source) of a specific parameter(s) is non-functional.

4.3 ERROR FRAME

The Error Frame contains a sequence of variable length dominant bits as a result of error flags being transmitted by different system-nodes. This is an important aspect of the 82526 communication protocol with regard to data consistency within a communication network. The error frame is followed by an error delimiter.

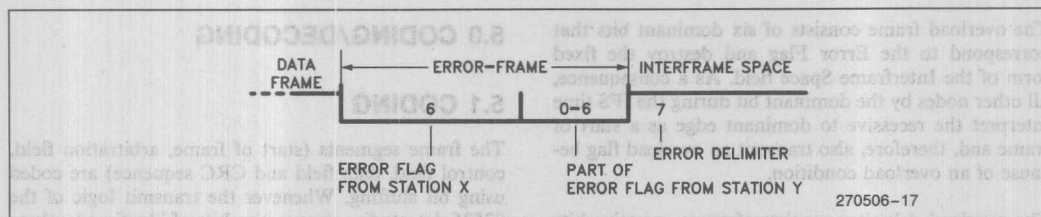


Figure 9. Error Frame Format

ERROR FLAG consists of six consecutive dominant bits. Since this "violates" bit stuffing guidelines, it is used as an error indicator to the system (see coding/decoding).

An Error Flag is transmitted if an 82526 operates as an error active node and has detected an error condition during or after a message transfer. A receiving or transmitting node may generate an error flag during or after a transmission. If an Error Flag is generated by a transmitter, or a receiver(s), all other nodes interpret the Error Flag as a bit stuffing rule violation. As a consequence, the transmission of an Error Flag occurs. The bit-sequence of dominant bits result from a superposition of different Error Flags transmitted by individual nodes. The total length of the Error Flag sequence varies between six bits minimum to twelve bits maximum.

An error condition is signaled by the transmission of six recessive bits while in the error passive operation mode. An error passive node with a temporary local receiver problem will not destroy messages received correctly by other nodes. The recessive bits may be overwritten by an Error Flag generated by one or more error active system nodes, but the error passive 82526 waits for at least six bits of equal polarity before entering into the next internal receive or transmit mode. (See Error Handling for error active/passive mode.)

NOTE:

The 82526 will not perform storage of an message (positive acceptance filtering) into the communication buffer, if reception of the message was followed by an Error Flag.

The error-delimiter consists of seven recessive bits generated by the 82526 after the end of an Error Flag on the serial bus line. This is monitored by detection of a transition from the dominant to recessive bit level.

Detected errors during the transmission of a data or remote frame can be signaled within the transmission time of the respective frame. This procedure associates an Error Flag to the corresponding frame, and initiates a retransmission of the frame. As the 82526 monitors, any deviation of its error frame, will start retransmitting an error frame if this occurs several times in a sequence the 82526 will become error passive.

4.4 OVERLOAD FRAME

The overload frame consists of two bit fields, the overload flag and the overload delimiter.

There are two cases of overload conditions which result in the transmission of an overload flag:

1. Internal conditions of the receiver circuitry of the 82526 which require a delay time before receiving the next frame (receiver not ready).
2. Detection of a "dominant" bit during Interframe Space.

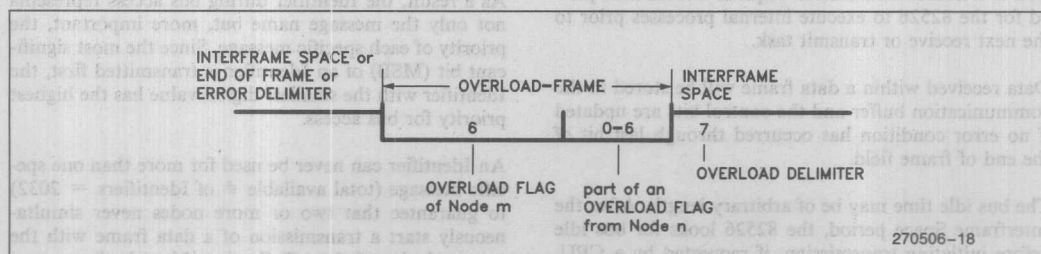


Figure 10. Overload Frame Format

The overload frame consists of six dominant bits that correspond to the Error Flag and destroy the fixed form of the Interframe Space field. As a consequence, all other nodes by the dominant bit during the IFS time interpret the recessive to dominant edge as a start of frame and, therefore, also transmit an overload flag because of an overload condition.

The overload delimiter consists of seven recessive bits generated by M&E 82526.

After transmission of an overload frame, each 82526 within the system monitors the bus line until a transition from a dominant to a recessive level occurs. This indicates to each 82526 the end of overload frames and each node simultaneously starts the transmission of six more recessive bits.

NOTE:

An overload frame can be transmitted earliest at the first bit time of the Interframe Space field. This is contrary to the Error Frame and allows the 82526 to differentiate between both.

4.6 INTERFRAME-SPACE

Data Frame and Remote Frame are separated from preceding frames by an Interframe Space consisting of the Intermission bit field and a possible Bus Idle time. An error frame is not preceded by an Interframe Space.

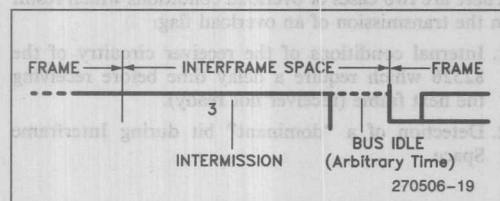


Figure 11. Interframe Space Format

INTERMISSION consists of three recessive bits. During Intermission time the 82526 will not start transmission of a frame. Intermission requires a fixed time period for the 82526 to execute internal processes prior to the next receive or transmit task.

Data received within a data frame will be stored in the communication buffer and the control bits are updated if no error condition has occurred through last bit of the end of frame field.

The bus idle time may be of arbitrary length. After the Interframe Space period, the 82526 looks for bus idle before initiating transmission, if requested by a CPU. The detection of a dominant bit after Intermission or bus idle is interpreted by the 82526 as Start of Frame.

5.0 CODING/DECODING

5.1 CODING

The frame segments (start of frame, arbitration field, control field, data field and CRC sequence) are coded using bit stuffing. Whenever the transmit logic of the 82526 detects five consecutive bits of identical levels to be transmitted, the logic inserts a complement bit in the transmitted bit stream.

5.2 DECODING

Whenever the 82526 has received five identical consecutive bit levels in the received bit stream the logic automatically deletes the next bit from the data stream (destuffing). For the bit fields with a fixed bit pattern the receiver destuffing logic of the 82526 is turned off.

6.0 ARBITRATION

In a case where two or more 82526s start a message transfer concurrently, the bus access conflict is solved by a bitwise arbitration method during transmission of the arbitration field.

The transmit logic compares the bit level transmitted to the level monitored on the serial bus. The transmit logic immediately will stop a current message transfer if a recessive bit was sent but a dominant bit was monitored. This method guarantees the data transfer of the message with the highest priority even if there is a collision during the arbitration field of one or more message Identifier(s).

The 82526 protocol architecture defines that each single message used in the communication network has a unique Identifier characterizing the type of data within the data field. Using this method, the Identifier assigns a name to the data frame and automatically implies the priority of the message.

As a result, the Identifier during bus access represents not only the message name but, more important, the priority of each specific message. Since the most significant bit (MSB) of an Identifier is transmitted first, the Identifier with the smallest digital value has the highest priority for bus access.

An Identifier can never be used for more than one specific message (total available # of Identifiers = 2032) to guarantee that two or more nodes never simultaneously start a transmission of a data frame with the same priority of data. Following this rule, bus access conflicts are resolved during the transmission of the

Identifier. One exception would be the simultaneous transmitter and receiver initiated frame transfer. If one 82526 generates a request for actual data of a certain type by transmitting a remote frame and simultaneously, the 82526 responsible for this type of data starts the transmission. Arbitration can not be solved by the Identifier itself.

For this reason, the RTR-bit is included in the arbitration field. The RTR-bit of the transmitter is always set dominant and, therefore, has a higher priority than the requesting 82526 (RTR-bit recessive). The remote frame request by the receiver gets an immediate response by the transmitter in this case.

7.0 ERROR DETECTION

The Error Detection mechanism is implemented for optimum error detection.

7.1 BIT ERROR

During a transmit operation, the 82526 monitors the bus on a bit-by-bit basis. If the bit level monitored is different from the transmitted one, a bit error is signaled.

Exceptions: Arbitration and ACK-SLOT. During arbitration, a recessive bit can be overwritten by a dominant bit. In this case, the 82526 interprets a bit error as an arbitration loss. During the ACK-SLOT, a transmitter may detect a falsified bit (recessive to dominant). This situation will only occur if all receivers have detected a CRC error and, therefore, this bit error will not be detected by 82526. This error is not critical because an error frame by the receivers will be generated after the ACK-SLOT.

NOTE:

Except during transmission of the arbitration field and during the time window of the ACK-SLOT, all global and local errors at the transmitter are detected.

7.2 BIT STUFFING ERROR

As described earlier, the frame segments are coded by a method of bit stuffing.

There are two possibilities where bit stuffing errors may occur:

1. A disturbance generates more consecutive bits of equal level than allowed by the rule of bit stuffing. These errors are detected by all nodes.

2. A disturbance falsifies one or more of the five bits proceeding the stuff bit. This error is not recognized by a receiver, but if an error appears at the transmitter as well, it will be detected as a bit error.

Otherwise, the error is detected by a receiver either by the bit stuffing mechanism (the stuff bit of the transmitter is not dropped but taken as an information bit) or by the CRC check.

7.3 CRC ERROR

To ensure the validity of a transmitted message, all receivers perform a CRC check. In addition to the information digits, any code word includes control digits used for error detection.

Description of the CRC Code

The code used for the 82526 is a (shortened) BCH Code, extended by a parity check and the following attributes:

- 127 bits as maximum length of the code word.
- 113 bits as maximum number of information digits (maximum 83 bits are used for the current implementation).
- length of the CRC sequence 15 bits.
- Hamming distance $d = 6$
 $d = \min A(x \text{ EXOR } y) / x, y \text{ different code words}$
 $A(x) = \text{number of "recessive" bits in the code word } x$

As a result, $d - 1$ random errors are detectable (some exceptions exist).

The CRC SEQUENCE is determined by the request that the code word, if interpreted as polynomial with coefficients 0 or 1 is divisible by the polynomial.

$$f(x) = (x^{14} + x^9 + x^6 + x^5 + x^4 + x^2 + x + 1)(x + 1) \\ = 1100011010011001$$

Burst errors are detected up to a length of 15 (degree of $f(x)$). Multiple errors (number of disturbed bits at least $d = 6$) are not detected with a residual error probability of 3×10^{-5} .

7.4 FORM ERROR

Form Errors result from the violation of the fixed form of the following bit fields:

- end of frame
- interframe space
- ACK delimiter
- CRC delimiter

During the transmission of these bit fields, an error condition is recognized if a "dominant" bit level is detected.

7.5 ERROR DETECTION CAPABILITIES

Global errors, which occur at all fully functional nodes, are 100% detectable.

For local errors, e.g. errors which may appear at some nodes only, the shortened BCH Code extended by the parity check has the following error detection capabilities:

- Up to 5 single bit errors are detected 100% even if those errors are being distributed randomly within the code word.
- All single bit errors are detected if their total number within the code word is odd.
- The residual error probability of the CRC check is $2^{-15} = 3 \times 10^{-5}$. As an error may be detected by the CRC check, and/or by additional implemented error detection mechanism, the residual error probability is significantly less than 3×10^{-5} .

7.6 ERROR CONFINEMENT

Definitions

OFF BUS

A node may enter the OFF-Bus mode initiated by the EML and will not receive or transmit any message until a reset request is sent by the host CPU, acknowledged by the IMP, and the RAM configuration routine executed and a predefined "wait time" (128x11 recessive bits) has passed.

ACK

A correctly received message is signaled to the transmitting node by setting a dominant bit level on the bus in the ACK-SLOT of the respective frame.

ERROR ACTIVE

Normal operation for each node is an error active mode. If an error is detected during transmission of a frame, a node enters into the send error flag state (see error frame).

An error passive node, like an error active node, may function as a receiver and/or transmitter, but actions based on transmit and/or receive error conditions are different. As an example, after detection of an error, an error passive node will send six recessive bits. If the error passive node is the transmitter of a disturbed frame, all other nodes detect an error with the six recessive bits since it violates the bit stuffing rule. An error passive receiver does not signal the detection of an error to the system and, therefore, will not care that error active nodes are not using a complete, correct message as received. An error passive node will, however, acknowledge the reception of a valid frame during the ACK-SLOT.

8.0 ADDRESS MAP OF THE 82526

The 82526 Address Map is shown in Figure 12:

ADDR.	
0	INTERNAL RAM AREA OF THE THE 82526
...	
3FH	
40H	DECODED BY CS0
...	
BFH	
C0H	DECODED BY CS1
...	
DFH	
E0H	DECODED BY CS2
...	
EFH	
FDH	CLOCK DIVIDER REGISTER
FEH	PORT 0 DATA REGISTER
FFH	PORT 1 DATA REGISTER

Figure 12. CAN Address Map

8.1 PORT DATA REGISTER

Port 0 and Port 1 of the 82526 are implemented in a way that allows the user to define the inputs and outputs of a port within the data itself. A written "1" to a port data register either means the appropriate pin is set high as an output function or this pin is configured as input. If a pin has been written to a "1", it may be read correctly.

	MSB	LSB							
ADD. 254	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	
ADD. 255	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	

8.2 CLOCK DIVIDER REGISTER

The 82526 provides a clock frequency (CLKOUT) to drive a CPU in the external clocked mode. CLKOUT is derived from the internal 82526 clock oscillator by a programmable divider register shown below:

	MSB				LSB			
ADD. 253	X	X	X	X	CCDR3	CCDR2	CCDR1	CCDR0

$$f_{\text{clkout}} = f_{\text{osc}} / \text{divider value}$$

$$\text{divider} = 2 \times (8 \times \text{CCDR3} + 4 \times \text{CCDR2} + 2 \times \text{CCDR1} + 1 \times \text{CCDR0} + 1)$$

9.0 BUS TIMING LOGIC (BTL)

The BTL monitors the serial bus line via its input comparator and performs all the bus line related bit timing. The baud rate prescaler also is a part of the BTL.

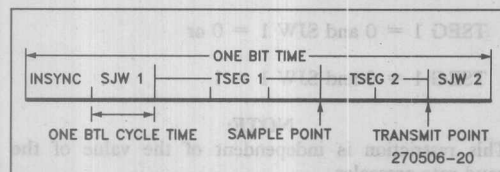
The BTL performs the following requirements:

- monitoring the serial bus level
- adjustment of the sample point within the bit time (programmable)
- synchronization to the bit stream
 - hard synchronization at start of frame bit
 - resynchronization during a transfer of a frame in order to compensate differences of transmitter/receiver clock frequencies of individual 82526s
- programmable majority logic
- prevention of nonsynchronization (spikes)

The configuration of the BTL is done during the initialization of the 82526. There are three registers in the 82526 communication buffer RAM which relate to the BTL: CONTROL-REGISTER, BUS-TIMING-REGISTER 0 and BUS-TIMING-REGISTER 1.

9.1 BIT TIMING

A bit time is subdivided in a number of BTL cycles. This number results from an addition of the programmable segments SJW 1, SJW 2, TSEG 1 and TSEG 2 plus the general segment INSYNC.



Meaning of the Segments

INSYNC

The incoming edge of a bit is expected during this state; this segment corresponds to one BTL cycle.

SJW 1/2 synchronization

Both segments determine the maximum SJW 2 jump width for resynchronization, programmable from 1 to 4 BTL cycles. The width of SJW 1 is increased to a maximum of two times the bit time during resynchronization. The width of SJW 2 is reduced or canceled to shorten the bit time during resynchronization.

TSEG 1

Determines the sampling point based on the number of BTL cycles programmed by TSEG 1 (4 bits). The sampling point is located at the end of TSEG 1 (SAM = 0). TSEG 1 is used to compensate delay times on the bus and to have some reserved time to tolerate one or more non-synchronization pulses caused by spikes on the bus line. TSEG 1 is programmable from 1 to 16 BTL cycles.

TSEG 2

Defines the time between the sampling point and the end of the bit time; programmable from 1 to 8 BTL cycles.

This segment is necessary to tolerate one or more non-synchronization spikes on the bus line. Also necessary to guarantee sufficient time for BSP to generate a transmit signal dependent on the sampled bus level. The transmit point is determined internally in such a way that with zero delay the generated transmit signal will appear within the INSYNC state. For example, no transmit signal would be generated if an arbitration was lost. This guarantees that the transmit logic immediately stops to continue a frame transfer and immediately enters into the receive mode.

Additional programmable segments are SAM, SYNCON and the baud rate. See control registers for details.

9.2 BIT TIME CALCULATION

The number of clock cycles at every bit time determines (together with the oscillator frequency and the baud-rate-prescaler) the period of each bit time and, as a consequence, the baud rate.

$$t_{\text{bit time}} = t_{\text{INSYNC}} + t_{\text{SJW 1}} + t_{\text{TSEG 1}} + t_{\text{TSEG 2}} + t_{\text{SJW 2}}$$

$$t_{\text{bit time}} = (2 \times \text{SJW} + \text{TSEG 1} + \text{TSEG 2} + 5) \times t_{\text{BTL cycle time}}$$

$$t_{\text{BTL cycle}} = 2 \times t_{\text{oscillator}} \times (\text{baud rate-prescaler} + 1)$$

$$\text{BAUD RATE} = \frac{2 \times f_{\text{oscillator}}}{2 \times n (\text{baud rate-prescaler} + 1) \times 4}$$

INSYNC = 1; n = number of BTL cycles for one bit time;

NOTE:

SJW 1, SJW 2, TSEG 1, TSEG 2, and the Baud rate prescaler are programmable numerical values.

9.3 REQUIREMENTS FOR CONFIGURATION OF THE BTL

Special requirements for the configuration of the BTL relate to the location of the sample point.

The correct location of the sample point is very important for proper function of a transmission, especially at high speed and maximum cable length. For this reason, the following items need to be considered:

- At start of frame, all 82526s in the system synchronize "hard" on the first recessive to dominant edge. During arbitration, however, more than one node may transmit in coincidence. As a result, it may take two times the bus delay plus the time of the output driver and the input comparator until the bus line is stable. Corresponding to this, the duration of TSEG 1 should reflect at least the total delay time.
- To improve the behavior with respect to spikes on the bus line, it is recommended to have an additional synchronization buffer on the left and right side of the sample point to allow one or more nonsynchronizations without sampling the wrong position within a bit time. At a minimum, this buffer should correspond to the time of the SJW segments.

9.4 CONFIGURATION RESTRICTIONS

A. Restriction related to the location of the transmit point.

Restrictions on the configuration are based on internal processing timing and relate to the location of the transmit point.

The transmit point is generated automatically, but the following conditions must be noted:

- After the sample point, the actual bus level is known. Dependent on the bus value, any node decides either to start, stop, or continue a transmission.

The decision requires a minimum of two BTL cycles for internal processing time. After this time period, the 82526 is ready to transmit.

- The start point to transmit a new bit occurs earlier than the bit actually monitored on the bus. This is to compensate the internal delay times such as synchronizing to the internal clock or to perform the majority process (SAM = 1).

Corresponding to the issues, the following restrictions regarding programming TSEG 2 and SJW1/SJW2 apply:

for SAM = 0: TSEG 2 + SJW 2 > = 3 clock cycles minimum

for SAM = 1: TSEG 2 + SJW 2 > = 4 clock cycles minimum

NOTE:

This restriction is valid only if the baud rate-prescaler is programmed to zero.

B. Restriction related to the Bit Stream Processor (BSP).

Because of internal processing time, it must be ensured that two successive sample points never get closer than a minimum of five BTL cycles even when shortened by hard synchronization or resynchronization. In the most critical case, the segments TSEG 2 and SJW 2 are shortened by a hard synchronization.

$$\text{SJW 1} + \text{TSEG 1} > = 4 \text{ BTL cycles min}$$

NOTE:

This restriction is valid only if the baud rate-prescaler is programmed to zero.

C. Restriction related to the Bit Timing Logic (BTL).

Because of BTL internal requirements, it is not allowed to program the following parameters at the same time:

$$\text{TSEG 1} = 0 \text{ and SJW 1} = 0 \text{ or}$$

$$\text{TSEG 1} = 0 \text{ and SJW 1} = 1$$

NOTE:

This restriction is independent of the value of the baud rate-prescaler.

9.5 MINIMUM CONFIGURATION OF BTL BAUD RATE-PRESCALER = 0

Considering all restrictions, a bit time can be programmed to a minimum of eight BTL cycles for baud rate-prescaler = 0 and SAM = 0.

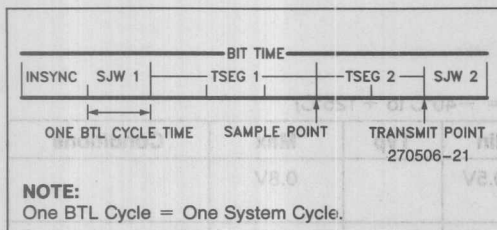


Figure 13. Minimum Configuration with Baud Rate-Prescaler = 0 and SAM = 0

Considering all restrictions, a bit time can be programmed to a minimum of six BTL cycles for baud-rate-prescaler ≥ 1 and SAM = 0.

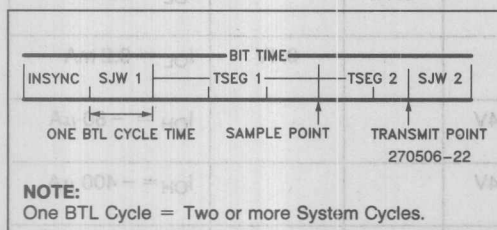


Figure 14. Minimum Configuration with Baud Rate-Prescaler = 0 and SAM = 0

10.0 SYNCHRONIZATION

Synchronization is performed by a state machine which compares the incoming edges with its actual bit timing and adapts the bit timing by hard synchronization or resynchronization.

HARD SYNCHRONIZATION occurs only at the beginning of a frame. The 82526 synchronizes to the first incoming recessive to dominant edge of a frame.

RESYNCHRONIZATION occurs during the message bit stream to compensate differences in the oscillator frequencies of individual 82526s as well as changes introduced by switching from one to another transmitter; e.g., after arbitration of two or more nodes.

SJW 1 and SJW 2 define the maximum number of clock cycles a bit time may be shortened or lengthened by resynchronization.

Resynchronization can be performed on both edges—recessive to dominant and dominant to recessive or on the recessive to dominant edge only, dependent on the programmed bit of SYNCON.

SYNCON = 1: Synchronization on both edges. Synchronization is always done on the edge of a bus level which is different from the one read at the last sample point.

SYNCON = 0: Synchronization on the edge of a dominant level only if the bus level monitored at the last sample point was a recessive level.

This restriction is necessary because of physical bus characteristics at the maximum cable length and high baud rates.

Synchronization is done once during a bit time and is released after the sample point dependent on the level of the actual bit. Thereafter, the BTL state machine synchronizes to the next relevant edge and then, synchronization is being suppressed until the following sample point.

By resynchronization, the bit time can be lengthened or shortened based on the following conditions:

- an edge appears in the segment SJW 1 or TSEG 1
→ lengthen
- an edge appears in the segment SJW 2 or TSEG 2
→ shorten

NOTE:

An 82526 transmitting a dominant bit does not lengthen (synchronize) the bit time if SYNCON is set to zero. This is important for a high speed bus in order to ensure the delay time of the output driver and input comparator does not cause a permanent lengthening of the bit time of the transmitter and, consequently, of all receivers.

11.0 ELECTRICAL CHARACTERISTICS

D. C. Characteristics ($V_{CC} = -4.5$ to $+5.5$ Volts; $T_C = -40^\circ\text{C}$ to $+125^\circ\text{C}$)

Symbol	Parameter	Min	Typ	Max	Conditions
V_{IL}	Input low voltage (All except XTAL1, XTAL2)	-0.5V		0.8V	
V_{IH}	Input high voltage (All except XTAL1, XTAL2, RESETBar)	2.0V		$V_{CC} + 0.5$	
V_{IH1}	Input high voltage (RESETBar)	3.5V		$V_{CC} + 0.5$	
	Hysteresis on RESETBar	200 mV			
V_{OL}	Output low voltage (All outputs except Port2, TX0, TX1)		0.4V		$I_{OL} = 1.6$ mA
V_{OL1}	Output low voltage (Port2)			0.4V	$I_{OL} = 3.2$ mA
V_{OH}	Output high voltage (All outputs except Port2, TX0, TX1)	2.4V			$I_{OH} = -80$ μA
V_{OH1}	Output high voltage (Port2)	2.4V			$I_{OH} = -400$ μA
I_{LK}	Input leakage current (except Port0/Port1)			± 10 μA	$V_{SS} < V_{IN} < V_{CC}$
I_{IL}	Low level input current (Port0/Port1)			-50 μA	$V_{IN} = 0.4$ V
I_{TL}	Logical 1 to 0 transition current (Port0/Port1)			-650 μA	$V_{IN} = 2$ V
I_{CC}	Supply current		22 mA	36 mA	fXTAL = 16 MHz
ISM	Sleep mode supply current		1.2 mA	2.2 mA	fXTAL = 1 MHz

A.C. CHARACTERISTICS

Conditions: $T_A = -40^\circ\text{C}$ to $+125^\circ\text{C}$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$, Port 2 outputs: $CL = 150\text{ pF}$, All other outputs: $CL = 80\text{ pF}$

Symbol	Parameter	Min	Typ	Max
1/tXTAL	Oscillator Frequency(1)	1 MHz		16 MHz
tAVLL	Address Valid to ALE Low	20 ns		
tLLAX	Address Hold after ALE Low	25 ns		
tLLDV	ALE or $\overline{CS}(2)$ Low to Valid Data Out	3tXTAL + 70 ns		5tXTAL + 70 ns
tRHDZ	Data Float after \overline{RD} High	10 ns		50 ns
tRLDV	Valid Data Out Delay from Read Control			70 ns
tQVWH	Input Data Setup to \overline{WR} High	20 ns		
tWHQX	Input Data Hold after \overline{WR} High	30 ns		
tWHDV	\overline{WR} High to Output Data Valid on Port 0 or Port 1			4tXTAL + 60 ns
tWHLL	\overline{WR} High to Next ALE or \overline{CS} Low(2, 3)	2tXTAL + 5 ns		
tWHWL	Time between Writes	4tXTAL + 5 ns		
tLLWH	ALE or $\overline{CS}(2)$ Low to \overline{WR} High	4tXTAL + 5 ns		
tLLYV	End of ALE to RDY Setup			100 ns
tLLYH	End of ALE to RDY High	4tXTAL + 50 ns		6tXTAL + 50 ns
tLCSL	ALE or \overline{CS} Low to $\overline{CS0-1-2}$ Low(2)			35 ns
tHCSX	$\overline{CS0-1-2}$ Float after ALE or \overline{CS} High(4, 5)			30 ns
tXINT	\overline{EXINT} Low to \overline{INT} Low			35 ns

NOTES:

1. The design goal is 24 MHz but this might not be possible with P445 process. 16 MHz is the value agreed upon by all parties during last design review.
2. Whichever falling edge is last.
3. Some pipe-lining of the write accesses is possible. This spec is important mainly when processor speed is higher than iCAN speed (use of the RDY output ...).
4. Whichever rising edge comes first.
5. $\overline{CS0}$, $\overline{CS1}$ and $\overline{CS2}$ are open drain outputs.

Comparator Specifications

RX0/RX1	Min	Max	Conditions
Input voltage	-0.5V	$V_{CC} + 0.5V$	
Common mode range	1.5V	$V_{CC} - 1.5V$	
Latchup trigger current	± 35 mA		

TX0/TX1	Min	Max	Conditions
Source current	-1.5 mA		$V_{out} = V_{CC} - 0.2V$
	-6.0 mA		$V_{out} = V_{CC} - 1.0V$
Sink current	5.0 mA		$V_{out} = 0.2V$
	20.0 mA		$V_{out} = 1.0V$
Maximum permitted source current (TX0, TX1 together)		-8.0 mA	
Maximum permitted sink current (TX0, TX1 together)		22.0 mA	

CLKOUT Specifications

CL = 50 pF	Min	Max	Conditions
CLKOUT frequency (1/T)	$f_{XTAL}/30^{(1)}$	16 MHz	
Rise time ⁽²⁾		15ns	
Fall time ⁽²⁾		10ns	
High time	20ns		$f_{XTAL} = 16$ MHz
Low time	20ns		$f_{XTAL} = 16$ MHz

NOTES:

1. $f_{OUT} = f_{XTAL}/R$ with $R = 1, 2, 4, 6, 8, \dots, 2 \times r$ and $r_{max} = 15$
2. Measured between 0.1 V_{CC} and 0.9 V_{CC}

MCS®-51 Architectural Overview

9

9

Overview MCS®-51 Architectural

ARCHITECTURAL OVERVIEW OF THE MCS®-51 FAMILY OF MICROCONTROLLERS

AUTOMOTIVE

MEMBERS OF THE FAMILY

The MCS®-51 family of microcontrollers consists of the devices listed in Table 1. The basic architectural structure of these devices is shown in Figure 1.

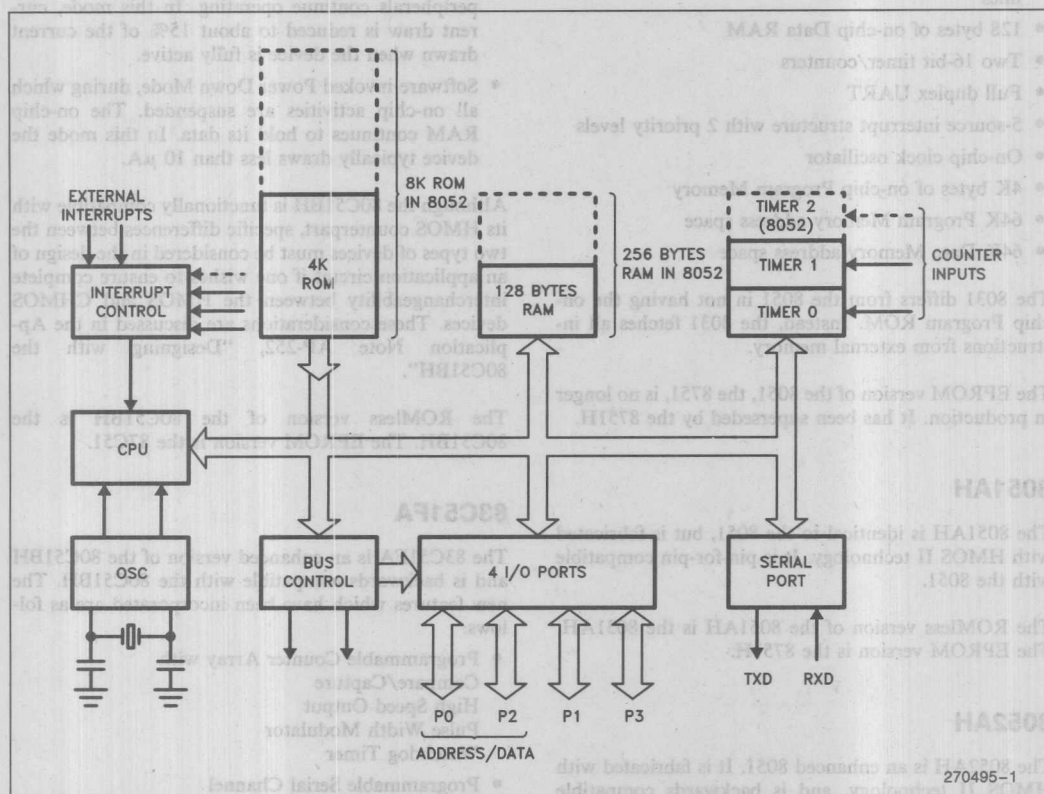


Figure 1. Block Diagram of the 8051/8052AH

Table 1. The MCS®-51 Family of Microcontrollers

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	16-Bit Timers	Ckt Type
8051	8031	(8751)	4K	128	2	HMOS
8051AH	8031AH	8751H	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS
83C51FA	80C51FA	87C51FA	8K	256	4	CHMOS
80C51GB	80C51GB	87C51GB	8K	256	4	CHMOS

8051

The 8051 is the original member of the MCS-51 Family, and has been in production since 1981. Among the features of the 8051 are:

- 8-bit CPU optimized for control applications
- Extensive Boolean processing (single-bit logic) capabilities
- 32 bidirectional and individually addressable I/O lines
- 128 bytes of on-chip Data RAM
- Two 16-bit timer/counters
- Full duplex UART
- 5-source interrupt structure with 2 priority levels
- On-chip clock oscillator
- 4K bytes of on-chip Program Memory
- 64K Program Memory address space
- 64K Data Memory address space

The 8031 differs from the 8051 in not having the on-chip Program ROM. Instead, the 8031 fetches all instructions from external memory.

The EPROM version of the 8051, the 8751, is no longer in production. It has been superseded by the 8751H.

8051AH

The 8051AH is identical to the 8051, but is fabricated with HMOS II technology. It is pin-for-pin compatible with the 8051.

The ROMless version of the 8051AH is the 8031AH. The EPROM version is the 8751H.

8052AH

The 8052AH is an enhanced 8051. It is fabricated with HMOS II technology, and is backwards compatible with the 8051. Its enhancements over the 8051 are as follows:

- 256 bytes of on-chip Data RAM
- Three timer/counters
- 6-source interrupt structure
- 8K bytes of on-chip Program ROM

The ROMless version of the 8052AH is the 8032AH. The EPROM version is the 8752BH.

A separate product, the 8052AH-BASIC, is an 8052AH with a full BASIC interpreter in the on-chip ROM.

80C51BH

The 80C51BH is the CHMOS version of the 8051. Functionally, it is fully compatible with the 8051, but being CMOS it draws less current than its HMOS counterpart. To further exploit the power savings available in CMOS circuitry, two reduced power modes are added:

- Software-invoked Idle Mode, during which the CPU is turned off while the RAM and other on-chip peripherals continue operating. In this mode, current draw is reduced to about 15% of the current drawn when the device is fully active.
- Software-invoked Power Down Mode, during which all on-chip activities are suspended. The on-chip RAM continues to hold its data. In this mode the device typically draws less than 10 μ A.

Although the 80C51BH is functionally compatible with its HMOS counterpart, specific differences between the two types of devices must be considered in the design of an application circuit if one wishes to ensure complete interchangeability between the HMOS and CHMOS devices. These considerations are discussed in the Application Note AP-252, "Designing with the 80C51BH".

The ROMless version of the 80C51BH is the 80C31BH. The EPROM version is the 87C51.

83C51FA

The 83C51FA is an enhanced version of the 80C51BH and is backwards compatible with the 80C51BH. The new features which have been incorporated are as follows:

- Programmable Counter Array with Compare/Capture High Speed Output Pulse Width Modulator Watchdog Timer
- Programmable Serial Channel Automatic Address Recognition Framing Error Detection
- Enhanced Power Down Mode
- Up/down timer/counter
- 8 Kbytes of on-chip Program ROM
- 256 bytes of on-chip Data RAM
- 7-source interrupt structure

For further information on these new features, refer to the "Hardware Description of the 83C51FA" chapter.

The ROMless version of the 83C51FA is the 80C51FA. The EPROM version is the 87C51FA.

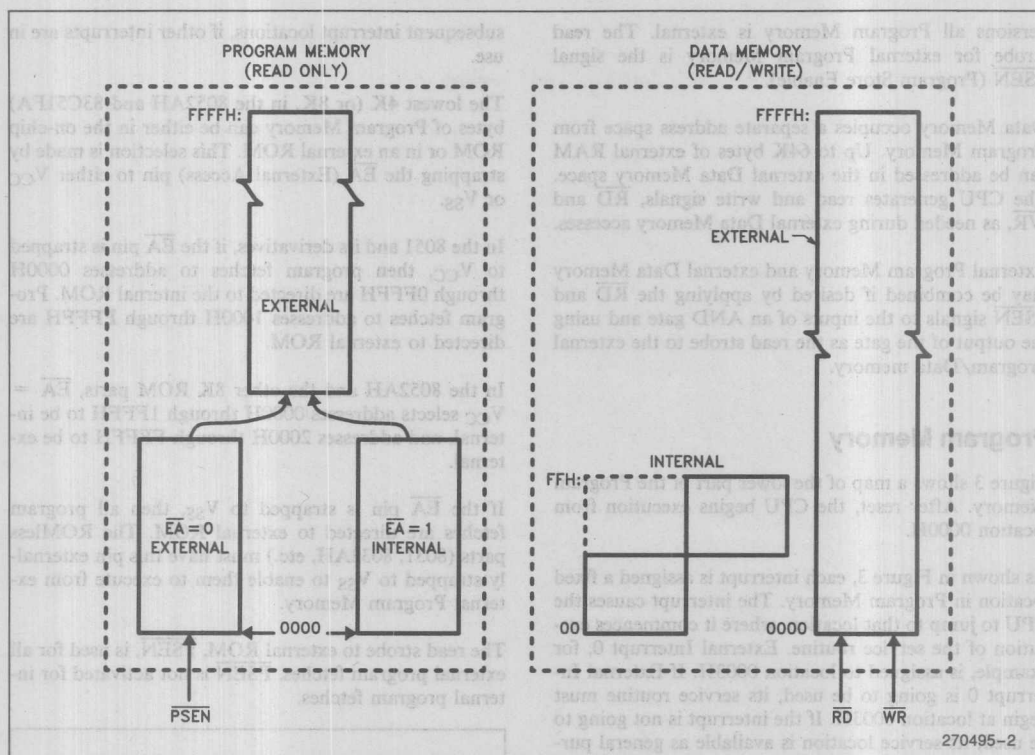


Figure 2. MCS®-51 Memory Structure

MEMORY ORGANIZATION IN MCS®-51 DEVICES

Logical Separation of Program and Data Memory

All MCS-51 devices have separate address spaces for Program and Data Memory, as shown in Figure 2. The logical separation of Program and Data Memory allows the Data Memory to be accessed by 8-bit addresses,

which can be more quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16-bit Data Memory addresses can also be generated through the DPTR register.

Program Memory can only be read, not written to. There can be up to 64K bytes of Program Memory. In the 8051, 8051AH, 80C51BH, and their EPROM versions, the lowest 4K bytes of Program Memory are on-chip. The 8052AH and 83C51FA provide 8 Kbytes of on-chip Program Memory storage. In the ROMless

versions all Program Memory is external. The read strobe for external Program Memory is the signal $\overline{\text{PSEN}}$ (Program Store Enable).

Data Memory occupies a separate address space from Program Memory. Up to 64K bytes of external RAM can be addressed in the external Data Memory space. The CPU generates read and write signals, $\overline{\text{RD}}$ and $\overline{\text{WR}}$, as needed during external Data Memory accesses.

External Program Memory and external Data Memory may be combined if desired by applying the $\overline{\text{RD}}$ and $\overline{\text{PSEN}}$ signals to the inputs of an AND gate and using the output of the gate as the read strobe to the external Program/Data memory.

Program Memory

Figure 3 shows a map of the lower part of the Program Memory. After reset, the CPU begins execution from location 0000H.

As shown in Figure 3, each interrupt is assigned a fixed location in Program Memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose Program Memory.

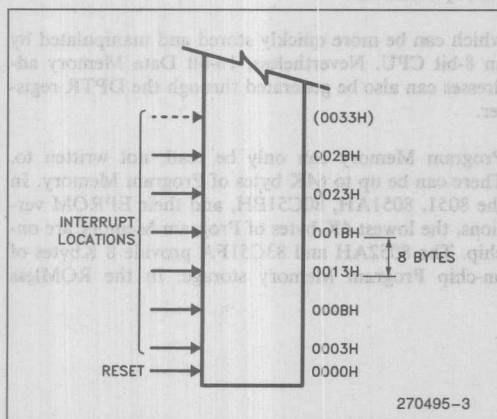


Figure 3. MCS®-51 Program Memory

The interrupt service locations are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1, etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over

subsequent interrupt locations, if other interrupts are in use.

The lowest 4K (or 8K, in the 8052AH and 83C51FA) bytes of Program Memory can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the $\overline{\text{EA}}$ (External Access) pin to either V_{CC} or V_{SS} .

In the 8051 and its derivatives, if the $\overline{\text{EA}}$ pin is strapped to V_{CC} , then program fetches to addresses 0000H through 0FFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.

In the 8052AH and the other 8K ROM parts, $\overline{\text{EA}} = \text{V}_{\text{CC}}$ selects addresses 0000H through 1FFFH to be internal, and addresses 2000H through FFFFH to be external.

If the $\overline{\text{EA}}$ pin is strapped to V_{SS} , then all program fetches are directed to external ROM. The ROMless parts (8031, 8032AH, etc.) must have this pin externally strapped to V_{SS} to enable them to execute from external Program Memory.

The read strobe to external ROM, $\overline{\text{PSEN}}$, is used for all external program fetches. $\overline{\text{PSEN}}$ is not activated for internal program fetches.

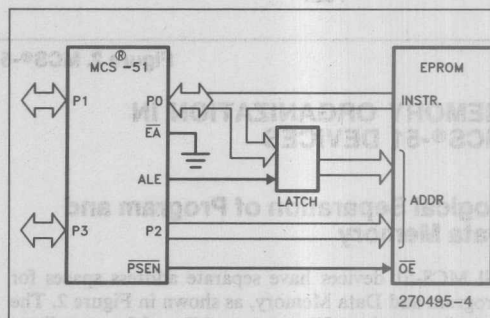


Figure 4. Executing from External Program Memory

The hardware configuration for external program execution is shown in Figure 4. Note that 16 I/O lines (Ports 0 and 2) are dedicated to bus functions during external Program Memory fetches. Port 0 (P0 in Figure 4) serves as a multiplexed address/data bus. It emits the low byte of the Program Counter (PCL) as an address, and then goes into a float state awaiting the arrival of the code byte from the Program Memory. During the time that the low byte of the Program Counter is valid on P0, the signal ALE (Address Latch Enable) clocks this byte into an address latch. Meanwhile, Port 2 (P2 in Figure 4) emits the high byte of the Program Counter (PCH). Then $\overline{\text{PSEN}}$ strobes the EPROM and the code byte is read into the microcontroller.

Program Memory addresses are always 16 bits wide, even though the actual amount of Program Memory used may be less than 64K bytes. External program execution sacrifices two of the 8-bit ports, P0 and P2, to the function of addressing the Program Memory.

Data Memory

The right half of Figure 2 shows the internal and external Data Memory spaces available to the MCS-51 user.

Figure 5 shows a hardware configuration for accessing up to 2K bytes of external RAM. The CPU in this case is executing from internal ROM. Port 0 serves as a multiplexed address/data bus to the RAM, and 3 lines of Port 2 are being used to page the RAM. The CPU generates RD and WR signals as needed during external RAM accesses.

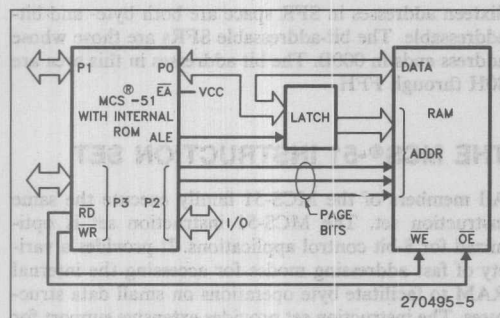


Figure 5. Accessing External Data Memory.
If the Program Memory is Internal, the Other Bits of P2 are Available as I/O.

There can be up to 64K bytes of external Data Memory. External Data Memory addresses can be either 1 or 2 bytes wide. One-byte addresses are often used in conjunction with one or more other I/O lines to page the RAM, as shown in Figure 5. Two-byte addresses can also be used, in which case the high address byte is emitted at Port 2.

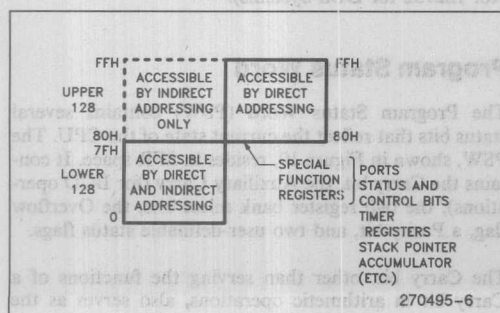


Figure 6. Internal Data Memory

Internal Data Memory is mapped in Figure 6. The memory space is shown divided into three blocks, which are generally referred to as the Lower 128, the Upper 128, and SFR space.

Internal Data Memory addresses are always one byte wide, which implies an address space of only 256 bytes. However, the addressing modes for internal RAM can in fact accommodate 384 bytes, using a simple trick. Direct addresses higher than 7FH access one memory space, and indirect addresses higher than 7FH access a different memory space. Thus Figure 6 shows the Upper 128 and SFR space occupying the same block of addresses, 80H through FFH, although they are physically separate entities.

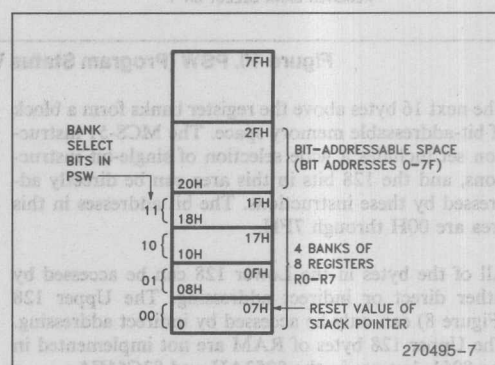


Figure 7. The Lower 128 Bytes of Internal RAM

The Lower 128 bytes of RAM are present in all MCS-51 devices as mapped in Figure 7. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

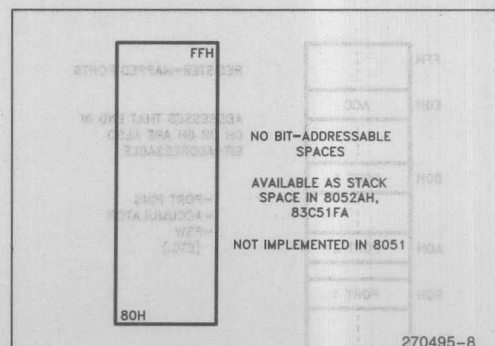


Figure 8. The Upper 128 Bytes of Internal RAM

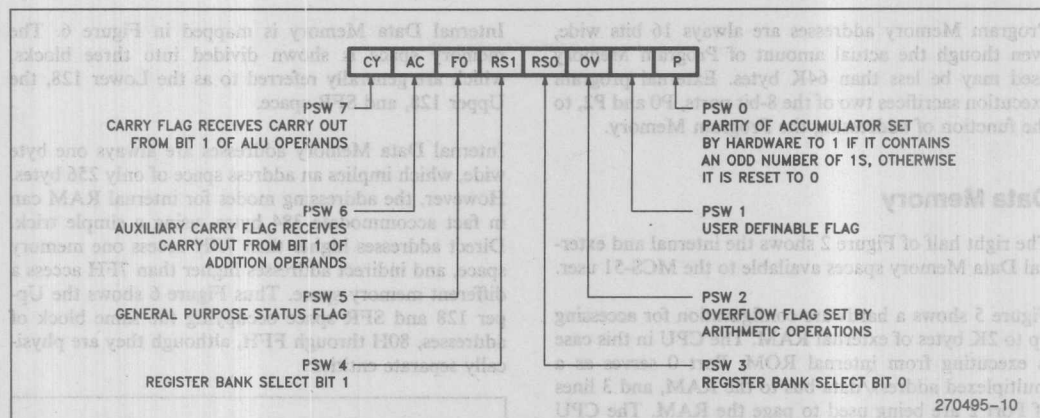


Figure 10. PSW (Program Status Word) Register in MCS®-51 Devices

The next 16 bytes above the register banks form a block of bit-addressable memory space. The MCS-51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing. The Upper 128 (Figure 8) can only be accessed by indirect addressing. The Upper 128 bytes of RAM are not implemented in the 8051, but are in the 8052AH and 83C51FA.

Figure 9 gives a brief look at the Special Function Register (SFR) space. SFRs include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. In general, all MCS-51 microcontrollers have the same SFRs as the 8051, and at the same addresses in SFR space. However, enhancements to the 8051 have additional SFRs that are not present in the 8051, nor perhaps in other proliferations of the family.

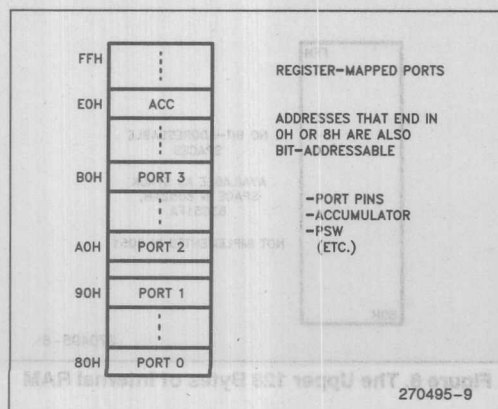


Figure 9. SFR Space

Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 000B. The bit addresses in this area are 80H through FFH.

THE MCS®-51 INSTRUCTION SET

All members of the MCS-51 family execute the same instruction set. The MCS-51 instruction set is optimized for 8-bit control applications. It provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. The instruction set provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that require Boolean processing.

An overview of the MCS-51 instruction set is presented below, with a brief description of how certain instructions might be used. References to "the assembler" in this discussion are to Intel's MCS-51 Macro Assembler, ASM51. More detailed information on the instruction set can be found in the MCS-51 Macro Assembler User's Guide (Order No. 9800937 for ISIS Systems, Order No. 122752 for DOS Systems).

Program Status Word

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown in Figure 10, resides in SFR space. It contains the Carry bit, the Auxiliary Carry (for BCD operations), the two register bank select bits, the Overflow flag, a Parity bit, and two user-definable status flags.

The Carry bit, other than serving the functions of a Carry bit in arithmetic operations, also serves as the "Accumulator" for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in Figure 7. A number of instructions refer to these RAM locations as R0 through R7. The selection of which of the four banks is being referred to is made on the basis of the bits RS0 and RS1 at execution time.

The Parity bit reflects the number of 1s in the Accumulator: $P = 1$ if the Accumulator contains an odd number of 1s, and $P = 0$ if the Accumulator contains an even number of 1s. Thus the number of 1s in the Accumulator plus P is always even.

Two bits in the PSW are uncommitted and may be used as general purpose status flags.

Addressing Modes

The addressing modes in the MCS-51 instruction set are as follows:

DIRECT ADDRESSING

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal Data RAM and SFRs can be directly addressed.

INDIRECT ADDRESSING

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected register bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit "data pointer" register, DPTR.

REGISTER INSTRUCTIONS

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits in the PSW.

REGISTER-SPECIFIC INSTRUCTIONS

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumulator as A assemble as accumulator-specific opcodes.

IMMEDIATE CONSTANTS

The value of a constant can follow the opcode in Program Memory. For example,

```
MOV A, #100
```

loads the Accumulator with the decimal number 100. The same number could be specified in hex digits as 64H.

INDEXED ADDRESSING

Only Program Memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in Program Memory. A 16-bit base register (either DPTR or the Program Counter) points to the base of the table, and the Accumulator is set up with the table entry number. The address of the table entry in Program Memory is formed by adding the Accumulator data to the base pointer.

Another type of indexed addressing is used in the "case jump" instruction. In this case the destination address of a jump instruction is computed as the sum of the base pointer and the Accumulator data.

Arithmetic Instructions

The menu of arithmetic instructions is listed in Table 2. The table indicates the addressing modes that can be used with each instruction to access the <byte> operand. For example, the ADD A, <byte> instruction can be written as:

```
ADD A, 7FH      (direct addressing)
ADD A, @R0      (indirect addressing)
ADD A, R7        (register addressing)
ADD A, #127     (immediate constant)
```

The execution times listed in Table 2 assume a 12 MHz clock frequency. All of the arithmetic instructions execute in 1 μ s except the INC DPTR instruction, which takes 2 μ s, and the Multiply and Divide instructions, which take 4 μ s.

Note that any byte in the internal Data Memory space can be incremented or decremented without going through the Accumulator.

One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory, so being able to increment it in one 16-bit operation is a useful feature.

The MUL AB instruction multiplies the Accumulator by the data in the B register and puts the 16-bit product into the concatenated B and Accumulator registers.

Table 2. A List of the MCS®-51 Arithmetic Instructions

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ADD A,<byte>	$A = A + \text{<byte>}$	X	X	X	X	1
ADDC A,<byte>	$A = A + \text{<byte>} + C$	X	X	X	X	1
SUBB A,<byte>	$A = A - \text{<byte>} - C$	X	X	X	X	1
INC A	$A = A + 1$	Accumulator only				1
INC <byte>	$\text{<byte>} = \text{<byte>} + 1$	X	X	X		1
INC DPTR	$\text{DPTR} = \text{DPTR} + 1$	Data Pointer only				2
DEC A	$A = A - 1$	Accumulator only				1
DEC <byte>	$\text{<byte>} = \text{<byte>} - 1$	X	X	X		1
MUL AB	$B:A = B \times A$	ACC and B only				4
DIV AB	$A = \text{Int}[A/B]$ $B = \text{Mod}[A/B]$	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

The DIV AB instruction divides the Accumulator by the data in the B register and leaves the 8-bit quotient in the Accumulator, and the 8-bit remainder in the B register.

Oddly enough, DIV AB finds less use in arithmetic “divide” routines than in radix conversions and programmable shift operations. An example of the use of DIV AB in a radix conversion will be given later. In shift operations, dividing a number by 2^n shifts its n bits to the right. Using DIV AB to perform the division

completes the shift in 4 μs and leaves the B register holding the bits that were shifted out.

The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD. Note that DA A will not convert a binary number to BCD. The DA A operation produces a meaningful result only as the second step in the addition of two BCD bytes.

Table 3. A List of the MCS®-51 Logical Instructions

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ANL A,<byte>	$A = A .\text{AND.} \text{<byte>}$	X	X	X	X	1
ANL <byte>,A	$\text{<byte>} = \text{<byte>} .\text{AND.} A$	X				1
ANL <byte>,#data	$\text{<byte>} = \text{<byte>} .\text{AND.} \text{\#data}$	X				2
ORL A,<byte>	$A = A .\text{OR.} \text{<byte>}$	X	X	X	X	1
ORL <byte>,A	$\text{<byte>} = \text{<byte>} .\text{OR.} A$	X				1
ORL <byte>,#data	$\text{<byte>} = \text{<byte>} .\text{OR.} \text{\#data}$	X				2
XRL A,<byte>	$A = A .\text{XOR.} \text{<byte>}$	X	X	X	X	1
XRL <byte>,A	$\text{<byte>} = \text{<byte>} .\text{XOR.} A$	X				1
XRL <byte>,#data	$\text{<byte>} = \text{<byte>} .\text{XOR.} \text{\#data}$	X				2
CRL A	$A = 00H$	Accumulator only				1
CPL A	$A = .\text{NOT.} A$	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

Logical Instructions

Table 3 shows the list of MCS-51 logical instructions. The instructions that perform Boolean operations (AND, OR, Exclusive OR, NOT) on bytes perform the operation on a bit-by-bit basis. That is, if the Accumulator contains 00110101B and <byte> contains 01010011B, then

ANL A,<byte>

will leave the Accumulator holding 00010001B.

The addressing modes that can be used to access the <byte> operand are listed in Table 3. Thus, the ANL A,<byte> instruction may take any of the forms

```
ANL A,7FH      (direct addressing)
ANL A,@R1      (indirect addressing)
ANL A,R6        (register addressing)
ANL A,#53H      (immediate constant)
```

All of the logical instructions that are Accumulator-specific execute in 1µs (using a 12 MHz clock). The others take 2 µs.

Note that Boolean operations can be performed on any byte in the internal Data Memory space without going through the Accumulator. The XRL <byte>, #data instruction, for example, offers a quick and easy way to invert port bits, as in

XRL P1,#0FFH

If the operation is in response to an interrupt, not using the Accumulator saves the time and effort to stack it in the service routine.

The Rotate instructions (RL A, RLC A, etc.) shift the Accumulator 1 bit to the left or right. For a left rotation, the MSB rolls into the LSB position. For a right rotation, the LSB rolls into the MSB position.

The SWAP A instruction interchanges the high and low nibbles within the Accumulator. This is a useful operation in BCD manipulations. For example, if the Accumulator contains a binary number which is known to be less than 100, it can be quickly converted to BCD by the following code:

```
MOV B,#10
DIV AB
SWAP A
ADD A,B
```

Dividing the number by 10 leaves the tens digit in the low nibble of the Accumulator, and the ones digit in the B register. The SWAP and ADD instructions move the tens digit to the high nibble of the Accumulator, and the ones digit to the low nibble.

Data Transfers

INTERNAL RAM

Table 4 shows the menu of instructions that are available for moving data around within the internal memory spaces, and the addressing modes that can be used with each one. With a 12 MHz clock, all of these instructions execute in either 1 or 2 µs.

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. Remember the Upper 128 bytes of data RAM can be accessed only by indirect addressing, and SFR space only by direct addressing.

Note that in all MCS-51 devices, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored,

Table 4. A List of the MCS®-51 Data Transfer Instructions that Access Internal Data Memory Space

Mnemonic	Operation	Addressing Modes				Execution Time (µs)
		Dir	Ind	Reg	Imm	
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>,<src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit immediate constant.				X	2
PUSH <src>	INC SP : MOV "@SP",<src>	X				2
POP <dest>	MOV <dest>,"@SP" : DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X			1

but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128, if they are implemented, but not into SFR space.

The Upper 128 are not implemented in the 8051, 8051AH, or 80C51BH, nor in their ROMless or EPROM counterparts. With these devices, if the SP points to the Upper 128, PUSHed bytes are lost, and POPped bytes are indeterminate.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory, or for 16-bit external Data Memory accesses.

The XCH A, <byte> instruction causes the Accumulator and addressed byte to exchange data. The XCHD A,@Ri instruction is similar, but only the low nibbles are involved in the exchange.

To see how XCH and XCHD can be used to facilitate data manipulations, consider first the problem of shifting an 8-digit BCD number two digits to the right. Figure 11 shows how this can be done using direct MOVs, and for comparison how it can be done using XCH instructions. To aid in understanding how the code works, the contents of the registers that are holding the BCD number and the content of the Accumulator are shown alongside each instruction to indicate their status after the instruction has been executed.

	2A	2B	2C	2D	2E	ACC
MOV A,2EH	00	12	34	56	78	78
MOV 2EH,2DH	00	12	34	56	56	78
MOV 2DH,2CH	00	12	34	34	56	78
MOV 2CH,2BH	00	12	12	34	56	78
MOV 2BH,#0	00	00	12	34	56	78

(a) Using direct MOVs: 14 bytes, 9 μ s

	2A	2B	2C	2D	2E	ACC
CLR A	00	12	34	56	78	00
XCH A,2BH	00	00	34	56	78	12
XCH A,2CH	00	00	12	56	78	34
XCH A,2DH	00	00	12	34	78	56
XCH A,2EH	00	00	12	34	56	78

(b) Using XCHs: 9 bytes, 5 μ s

Figure 11. Shifting a BCD Number Two Digits to the Right

	2A	2B	2C	2D	2E	ACC
MOV A,2EH	00	12	34	56	78	78
MOV 2EH,2DH	00	12	34	56	56	78
MOV 2DH,2CH	00	12	34	34	56	78
MOV 2CH,2BH	00	12	12	34	56	78
MOV 2BH,#0	00	00	12	34	56	78

(a) Using direct MOVs: 14 bytes, 9 μ s

	2A	2B	2C	2D	2E	ACC
CLR A	00	12	34	56	78	00
XCH A,2BH	00	00	34	56	78	12
XCH A,2CH	00	00	12	56	78	34
XCH A,2DH	00	00	12	34	78	56
XCH A,2EH	00	00	12	34	56	78

(b) Using XCHs: 9 bytes, 5 μ s

After the routine has been executed, the Accumulator contains the two digits that were shifted out on the right. Doing the routine with direct MOVs uses 14 code bytes and 9 μ s of execution time (assuming a 12 MHz clock). The same operation with XCHs uses less code and executes almost twice as fast.

To right-shift by an odd number of digits, a one-digit shift must be executed. Figure 12 shows a sample of code that will right-shift a BCD number one digit, using the XCHD instruction. Again, the contents of the registers holding the number and of the Accumulator are shown alongside each instruction.

	2A	2B	2C	2D	2E	ACC
MOV R1,#2EH	00	12	34	56	78	XX
MOV R0,#2DH	00	12	34	56	78	XX
loop for R1 = 2EH:						
LOOP: MOV A,@R1	00	12	34	56	78	78
XCHD A,@R0	00	12	34	58	78	76
SWAP A	00	12	34	58	78	67
MOV @R1,A	00	12	34	58	67	67
DEC R1	00	12	34	58	67	67
DEC R0	00	12	34	58	67	67
CJNE R1,#2AH,LOOP						
loop for R1 = 2DH:	00	12	38	45	67	45
loop for R1 = 2CH:	00	18	23	45	67	23
loop for R1 = 2BH:	08	01	23	45	67	01
CLR A	08	01	23	45	67	00
XCH A,2AH	00	01	23	45	67	08

Figure 12. Shifting a BCD Number One Digit to the Right

First, pointers R1 and R0 are set up to point to the two bytes containing the last four BCD digits. Then a loop is executed which leaves the last byte, location 2EH, holding the last two digits of the shifted number. The pointers are decremented, and the loop is repeated for location 2DH. The CJNE instruction (Compare and Jump if Not Equal) is a loop control that will be described later.

The loop is executed from LOOP to CJNE for R1 = 2EH, 2DH, 2CH and 2BH. At that point the digit that was originally shifted out on the right has propagated to location 2AH. Since that location should be left with 0s, the lost digit is moved to the Accumulator.

EXTERNAL RAM

Table 5 shows a list of the Data Transfer instructions that access external Data Memory. Only indirect addressing can be used. The choice is whether to use a one-byte address, @Ri, where Ri can be either R0 or R1 of the selected register bank, or a two-byte address, @DPTR. The disadvantage to using 16-bit addresses if only a few K bytes of external RAM are involved is that 16-bit addresses use all 8 bits of Port 2 as address bus. On the other hand, 8-bit addresses allow one to address a few K bytes of RAM, as shown in Figure 5, without having to sacrifice all of Port 2.

All of these instructions execute in 2 μ s, with a 12 MHz clock.

Table 5. A List of the MCS®-51 Data Transfer Instructions that Access External Data Memory Space

Address Width	Mnemonic	Operation	Execution Time (μ s)
8 bits	MOVX A, @Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri, A	Write external RAM @Ri	2
16 bits	MOVX A, @DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR, A	Write external RAM @DPTR	2

Note that in all external Data RAM accesses, the Accumulator is always either the destination or source of the data.

The read and write strobes to external RAM are activated only during the execution of a MOVX instruction. Normally these signals are inactive, and in fact if they're not going to be used at all, their pins are available as extra I/O lines. More about that later.

LOOKUP TABLES

Table 6 shows the two instructions that are available for reading lookup tables in Program Memory. Since these instructions access only Program Memory, the lookup tables can only be read, not updated. The mnemonic is MOVC for "move constant".

If the table access is to external Program Memory, then the read strobe is $\overline{\text{PSEN}}$.

Table 6. The MCS®-51 Lookup Table Read Instructions

Mnemonic	Operation	Execution Time (μ s)
MOVC A, @A+DPTR	Read Pgm Memory at (A+DPTR)	2
MOVC A, @A+PC	Read Pgm Memory at (A+PC)	2

The first MOVC instruction in Table 6 can accommodate a table of up to 256 entries, numbered 0 through 255. The number of the desired entry is loaded into the Accumulator, and the Data Pointer is set up to point to beginning of the table. Then

```
MOVC A, @A+DPTR
```

copies the desired table entry into the Accumulator.

The other MOVC instruction works the same way, except the Program Counter (PC) is used as the table base, and the table is accessed through a subroutine. First the number of the desired entry is loaded into the Accumulator, and the subroutine is called:

```
MOV A, ENTRY_NUMBER
CALL TABLE
```

The subroutine "TABLE" would look like this:

```
TABLE: MOVC A, @A+PC
RET
```

The table itself immediately follows the RET (return) instruction in Program Memory. This type of table can have up to 255 entries, numbered 1 through 255. Number 0 can not be used, because at the time the MOVC instruction is executed, the PC contains the address of the RET instruction. An entry numbered 0 would be the RET opcode itself.

Boolean Instructions

MCS-51 devices contain a complete Boolean (single-bit) processor. The internal RAM contains 128 addressable bits, and the SFR space can support up to 128 other addressable bits. All of the port lines are bit-addressable, and each one can be treated as a separate single-bit port. The instructions that access these bits are not just conditional branches, but a complete menu of move, set, clear, complement, OR, and AND instructions. These kinds of bit operations are not easily obtained in other architectures with any amount of byte-oriented software.

Table 7. A List of the MCS®-51 Boolean Instructions

Mnemonic	Operation	Execution Time (μs)
ANL C,bit	C = C.AND. bit	2
ANL C,/bit	C = C.AND. .NOT. bit	2
ORL C,bit	C = C.OR. bit	2
ORL C,/bit	C = C.OR. .NOT. bit	2
MOV C,bit	C = bit	1
MOV bit,C	bit = C	2
CLR C	C = 0	1
CLR bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = .NOT. C	1
CPL bit	bit = .NOT. bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit,rel	Jump if bit = 1	2
JNB bit,rel	Jump if bit = 0	2
JBC bit,rel	Jump if bit = 1; CLR bit	2

The instruction set for the Boolean processor is shown in Table 7. All bit accesses are by direct addressing. Bit addresses 00H through 7FH are in the Lower 128, and bit addresses 80H through FFH are in SFR space.

Note how easily an internal flag can be moved to a port pin:

```

MOV C,FLAG
MOV P1.0,C

```

In this example, FLAG is the name of any addressable bit in the Lower 128 or SFR space. An I/O line (the LSB of Port 1, in this case) is set or cleared depending on whether the flag bit is 1 or 0.

The Carry bit in the PSW is used as the single-bit Accumulator of the Boolean processor. Bit instructions that refer to the Carry bit as C assemble as Carry-specific instructions (CLR C, etc). The Carry bit also has a direct address, since it resides in the PSW register, which is bit-addressable.

Note that the Boolean instruction set includes ANL and ORL operations, but not the XRL (Exclusive OR) operation. An XRL operation is simple to implement in software. Suppose, for example, it is required to form the Exclusive OR of two bits:

```

C = bit1.XRL. bit2

```

The software to do that could be as follows:

```

MOV C,bit1
JNB bit2,OVER
CPL C

```

OVER: (continue)

First, bit1 is moved to the Carry. If bit2 = 0, then C now contains the correct result. That is, bit1.XRL. bit2 = bit1 if bit2 = 0. On the other hand, if bit2 = 1 C now contains the complement of the correct result. It need only be inverted (CPL C) to complete the operation.

This code uses the JNB instruction, one of a series of bit-test instructions which execute a jump if the addressed bit is set (JC, JB, JBC) or if the addressed bit is not set (JNC, JNB). In the above case, bit2 is being tested, and if bit2 = 0 the CPL C instruction is jumped over.

JBC executes the jump if the addressed bit is set, and also clears the bit. Thus a flag can be tested and cleared in one operation.

All the PSW bits are directly addressable, so the Parity bit, or the general purpose flags, for example, are also available to the bit-test instructions.

RELATIVE OFFSET

The destination address for these jumps is specified to the assembler by a label or by an actual address in Program Memory. However, the destination address assembles to a **relative** offset byte. This is a signed (two's complement) offset byte which is added to the PC in two's complement arithmetic if the jump is executed.

The range of the jump is therefore -128 to +127 Program Memory bytes relative to the first byte following the instruction.

Jump Instructions

Table 8 shows the list of unconditional jumps.

Table 8. Unconditional Jumps in MCS®-51 Devices

Mnemonic	Operation	Execution Time (μs)
JMP addr	Jump to addr	2
JMP @A + DPTR	Jump to A + DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

The Table lists a single "JMP addr" instruction, but in fact there are three—SJMP, LJMP and AJMP—which differ in the format of the destination address. JMP is a generic mnemonic which can be used if the programmer does not care which way the jump is encoded.

The SJMP instruction encodes the destination address as a relative offset, as described above. The instruction is 2 bytes long, consisting of the opcode and the relative offset byte. The jump distance is limited to a range of -128 to +127 bytes relative to the instruction following the SJMP.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2K block as the instruction following the AJMP.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2K block as the instruction following the AJMP.

In all cases the programmer specifies the destination address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the destination address into the correct format for the given instruction. If the format required by the instruction will not support the distance to the specified destination address, a "Destination out of range" message is written into the List file.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and

the Accumulator. Typically, DPTR is set up with the address of a jump table, and the Accumulator is given an index to the table. In a 5-way branch, for example, an integer 0 through 4 is loaded into the Accumulator. The code to be executed might be as follows:

```
MOV DPTR, #JUMP_TABLE
MOV A, INDEX_NUMBER
RL A
JMP @A + DPTR
```

The RL A instruction converts the index number (0 through 4) to an even number on the range 0 through 8, because each entry in the jump table is 2 bytes long:

JUMP_TABLE:

```
AJMP CASE_0
AJMP CASE_1
AJMP CASE_2
AJMP CASE_3
AJMP CASE_4
```

Table 8 shows a single "CALL addr" instruction, but there are two of them—LCALL and ACALL—which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 64K Program Memory space. The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2K block as the instruction following the ACALL.

In any case the programmer specifies the subroutine address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the address into the correct format for the given instructions.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done. If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 9 shows the list of conditional jumps available to the MCS-51 user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of -128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

Table 9. Conditional Jumps in MCS®-51 Devices

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
JZ rel	Jump if A = 0					2
JNZ rel	Jump if A ≠ 0					2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

There is no Zero bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control. To execute a loop N times, load a counter byte with N and terminate the loop with a DJNZ to the beginning of the loop, as shown below for N = 10:

```

MOV     COUNTER,#10
LOOP:   (begin loop)
        *
        *
        *
        (end loop)
        DJNZ COUNTER,LOOP
        (continue)

```

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control as in Figure 12. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. In the example of Figure 12, the two bytes were the data in R1 and the constant 2AH. The initial data in R1 was 2EH. Every time the loop was executed, R1 was decremented, and the looping was to continue until the R1 data reached 2AH.

Another application of this instruction is in "greater than, less than" comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry bit is set (1). If the first is greater than or equal to the second, then the Carry bit is cleared.

CPU TIMING

All MCS-51 microcontrollers have an on-chip oscillator which can be used if desired as the clock source for the CPU. To use the on-chip oscillator, connect a crystal or ceramic resonator between the XTAL1 and XTAL2 pins of the microcontroller, and capacitors to ground as shown in Figure 13.

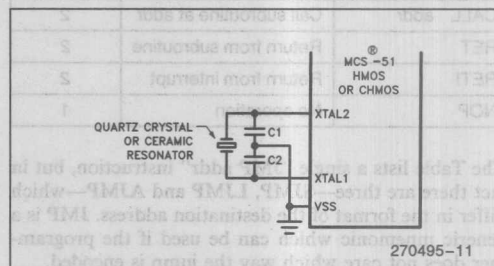
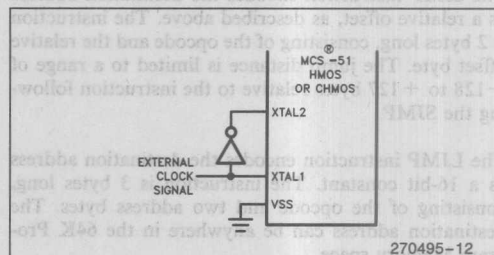
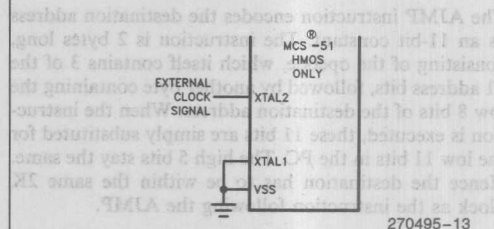


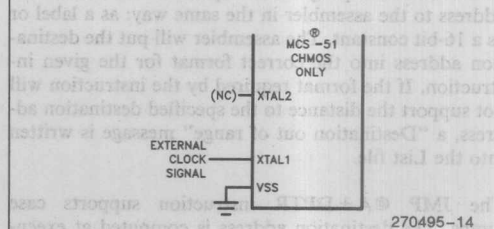
Figure 13. Using the On-Chip Oscillator



A. H MOS or CH MOS



B. H MOS Only



C. CH MOS Only

Figure 14. Using an External Clock

Examples of how to drive the clock with an external oscillator are shown in Figure 14. Note that in the HMOS devices (8051, etc.) the signal at the XTAL2 pin actually drives the internal clock generator. In the CHMOS devices (80C51BH, etc.) the signal at the XTAL1 pin drives the internal clock generator. If only one pin is going to be driven with the external oscillator signal, make sure it is the right pin.

The internal clock generator defines the sequence of states that make up the MCS-51 machine cycle.

Machine Cycles

A machine cycle consists of a sequence of 6 states, numbered S1 through S6. Each state time lasts for two oscillator periods. Thus a machine cycle takes 12 oscillator periods or $1 \mu\text{s}$ if the oscillator frequency is 12 MHz.

Each state is divided into a Phase 1 half and a Phase 2 half. Figure 15 shows the fetch/execute sequences in

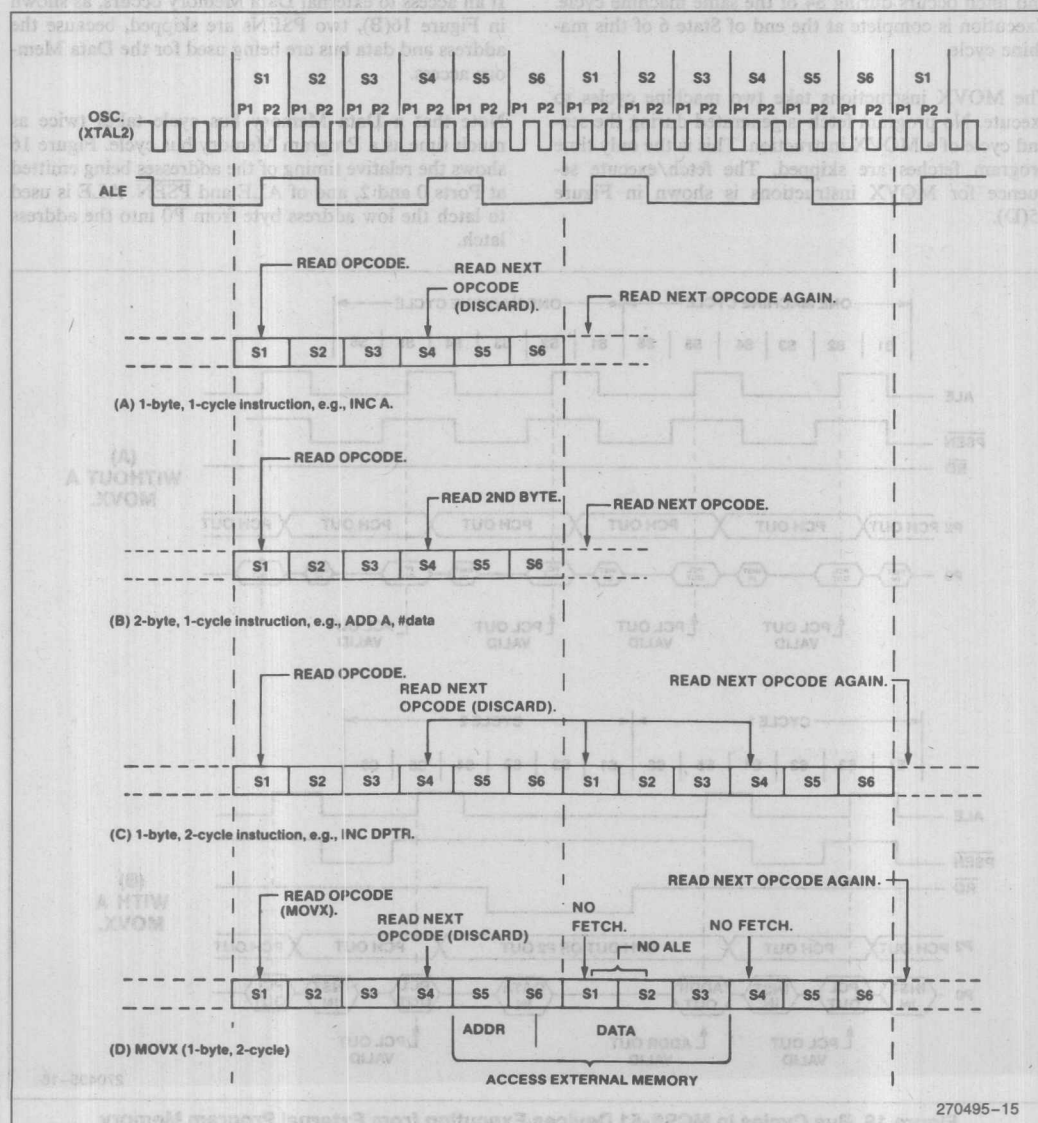


Figure 15. State Sequences in MCS-51 Devices

states and phases for various kinds of instructions. Normally two program fetches are generated during each machine cycle, even if the instruction being executed doesn't need more code bytes, the CPU simply ignores the extra fetch, and the Program Counter is not incremented.

Execution of a one-cycle instruction (Figure 15A and B) begins during State 1 of the machine cycle, when the opcode is latched into the Instruction Register. A second fetch occurs during S4 of the same machine cycle. Execution is complete at the end of State 6 of this machine cycle.

The MOVX instructions take two machine cycles to execute. No program fetch is generated during the second cycle of a MOVX instruction. This is the only time program fetches are skipped. The fetch/execute sequence for MOVX instructions is shown in Figure 15(D).

The fetch/execute sequences are the same whether the Program Memory is internal or external to the chip. Execution times do not depend on whether the Program Memory is internal or external.

Figure 16 shows the signals and timing involved in program fetches when the Program Memory is external. If Program Memory is external, then the Program Memory read strobe $\overline{\text{PSEN}}$ is normally activated twice per machine cycle, as shown in Figure 16(A).

If an access to external Data Memory occurs, as shown in Figure 16(B), two $\overline{\text{PSEN}}$ s are skipped, because the address and data bus are being used for the Data Memory access.

Note that a Data Memory bus cycle takes twice as much time as a Program Memory bus cycle. Figure 16 shows the relative timing of the addresses being emitted at Ports 0 and 2, and of ALE and $\overline{\text{PSEN}}$. ALE is used to latch the low address byte from P0 into the address latch.

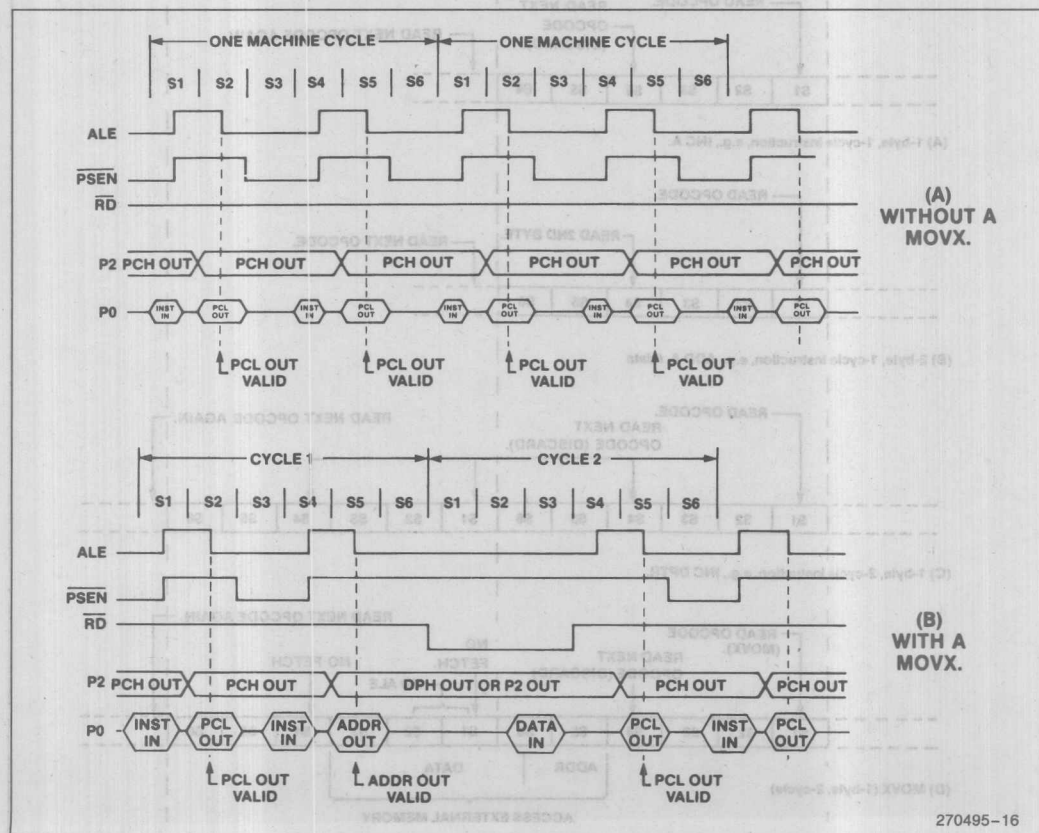


Figure 16. Bus Cycles in MCS®-51 Devices Executing from External Program Memory

When the CPU is executing from internal Program Memory, PSEN is not activated, and program addresses are not emitted. However, ALE continues to be activated twice per machine cycle and so is available as a clock output signal. Note, however, that one ALE is skipped during the execution of the MOVX instruction.

Interrupt Structure

The 8051, 8051AH, and 80C51BH, and their ROMless and EPROM versions, provide 5 interrupt sources: 2 external interrupts, 2 timer interrupts, and the serial port interrupt. The 8052AH provides these 5 plus a sixth interrupt that is associated with the third timer/counter which is present in this device. Additional interrupts are available on the 83C51FA. Refer to the appropriate chapters on these devices for further information on their interrupts.

What follows is an overview of the interrupt structure for these devices. More detailed information for specific members of the MCS-51 family is provided in the chapters of this handbook that describe the specific devices.

INTERRUPT ENABLES

Each of the interrupt sources can be individually enabled or disabled by setting or clearing a bit in the SFR

(MSB)				(LSB)			
EA	—	ET2	ES	ET1	EX1	ET0	EX0
Symbol	Position	Function					
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.					
—	IE.6	reserved					
ET2	IE.5	enables or disables the Timer 2 overflow or capture interrupt. If ET2 = 0, the Timer 2 interrupt is disabled.					
ES	IE.4	enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.					
ET1	IE.3	enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.					
EX1	IE.2	enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.					
ET0	IE.1	enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.					
EX0	IE.0	enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.					

Figure 17. IE (Interrupt Enable) Register in the 8052AH

named IE (Interrupt Enable). This register also contains a global disable bit, which can be cleared to disable all interrupts at once. Figure 17 shows the IE register for the 8052AH.

INTERRUPT PRIORITIES

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in the SFR named IP (Interrupt Priority). Figure 18 shows the IP register in the 8052AH.

A low-priority interrupt can be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence.

Figure 19 shows, for the 8052AH, how the IE and IP registers and the polling sequence work to determine which if any interrupt will be serviced.

(MSB)				(LSB)			
—	—	PT2	PS	PT1	PX1	PT0	PX0
Symbol	Position	Function					
—	IP.7	reserved					
—	IP.6	reserved					
PT2	IP.5	defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the higher priority level.					
PS	IP.4	defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.					
PT1	IP.3	defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.					
PX1	IP.2	defines the External Interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.					
PT0	IP.1	defines the Timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.					
PX0	IP.0	defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.					

Figure 18. IP (Interrupt Priority) Register in the 8052AH

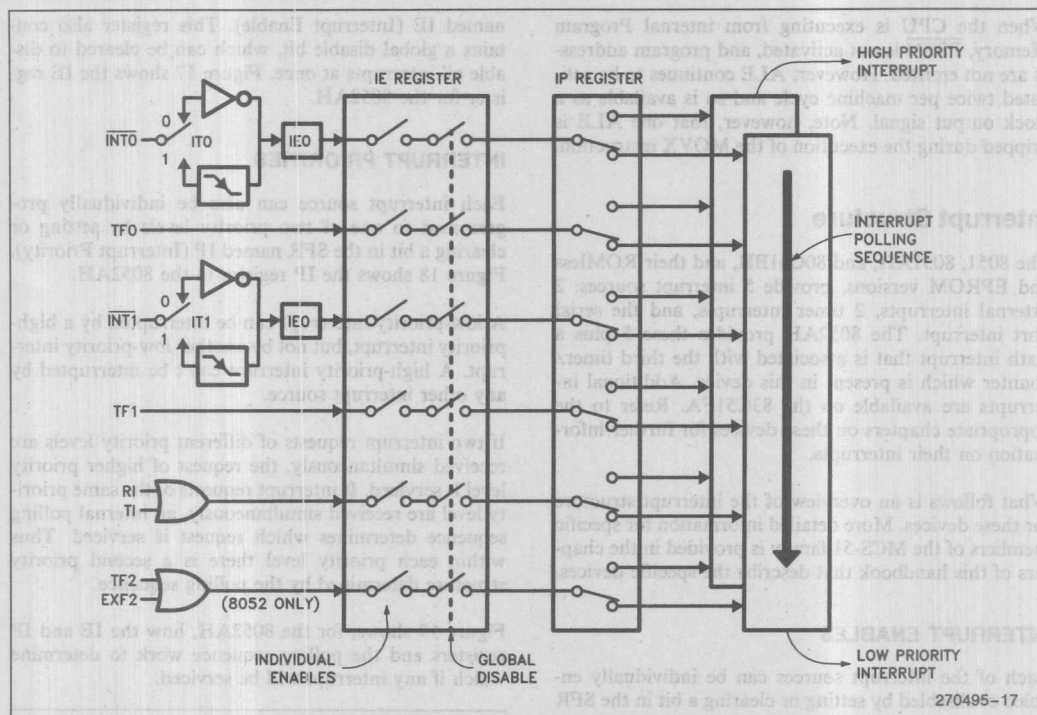


Figure 19. 8052 Interrupt Control System

In operation, all the interrupt flags are latched into the interrupt control system during State 5 of every machine cycle. The samples are polled during the following machine cycle. If the flag for an enabled interrupt is found to be set (1), the interrupt system generates an LCALL to the appropriate location in Program Memory, unless some other condition blocks the interrupt. Several conditions can block an interrupt, among them that an interrupt of equal or higher priority level is already in progress.

The hardware-generated LCALL causes the contents of the Program Counter to be pushed onto the stack, and reloads the PC with the beginning address of the service routine. As previously noted (Figure 3), the service routine for each interrupt begins at a fixed location.

Only the Program Counter is automatically pushed onto the stack, not the PSW or any other register. Having only the PC be automatically saved allows the programmer to decide how much time to spend saving which other registers. This enhances the interrupt response time, albeit at the expense of increasing the programmer's burden of responsibility. As a result, many interrupt functions that are typical in control applications—toggling a port pin, for example, or reloading a timer, or unloading a serial buffer—can often be com-

pleted in less time than it takes other architectures to commence them.

SIMULATING A THIRD PRIORITY LEVEL IN SOFTWARE

Some applications require more than the two priority levels that are provided by on-chip hardware in MCS-51 devices. In these cases, relatively simple software can be written to produce the same effect as a third priority level.

First, interrupts that are to have higher priority than 1 are assigned to priority 1 in the IP (Interrupt Priority) register. The service routines for priority 1 interrupts that are supposed to be interruptible by "priority 2" interrupts are written to include the following code:

```
PUSH    IE
MOV     IE,#MASK
CALL    LABEL
*****
(execute service routine)
*****
POP     IE
RET
LABEL: RETI
```

As soon as any priority 1 interrupt is acknowledged, the IE (Interrupt Enable) register is re-defined so as to disable all but "priority 2" interrupts. Then, a CALL to LABEL executes the RETI instruction, which clears the priority 1 interrupt-in-progress flip-flop. At this point any priority 1 interrupt that is enabled can be serviced, but only "priority 2" interrupts are enabled.

POPping IE restores the original enable byte. Then a normal RET (rather than another RETI) is used to terminate the service routine. The additional software adds 10 μ s (at 12 MHz) to priority 1 interrupts.

8051, 8052 and 80C51

10

HARDWARE DESCRIPTION OF THE 8051, 8052 AND 80C51 AUTOMOTIVE

INTRODUCTION

This chapter presents a comprehensive description of the on-chip hardware features of the MCS[®]-51 micro-controllers. Included in this description are

- The port drivers and how they function both as ports and, for Ports 0 and 2, in bus operations
- The Timer/Counters
- The Serial Interface
- The Interrupt System
- Reset
- The Reduced Power Modes in the CHMOS devices

- The EPROM versions of the 8051AH, 8052AH, and 80C51BH

The devices under consideration are listed in Table 1. As it becomes unwieldy to be constantly referring to each of these devices by their individual names, we will adopt a convention of referring to them generically as 8051s and 8052s, unless a specific member of the group is being referred to, in which case it will be specifically named. The "8051s" include the 8051, 8051AH, and 80C51BH, and their ROMless and EPROM versions. The "8052s" are the 8052AH, 8032AH, and 8752BH.

Figure 1 shows a functional block diagram of the 8051s and 8052s.

Table 1. The MCS-51 Family of Microcontrollers

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	16-bit Timers	Ckt Type
8051	8031	(8751)	4K	128	2	HMOS
8051AH	8031AH	8751H	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS

Special Function Registers

A map of the on-chip memory area called SFR (Special Function Register) space is shown in Figure 2. SFRs marked by parentheses are resident in the 8052s but not in the 8051s.

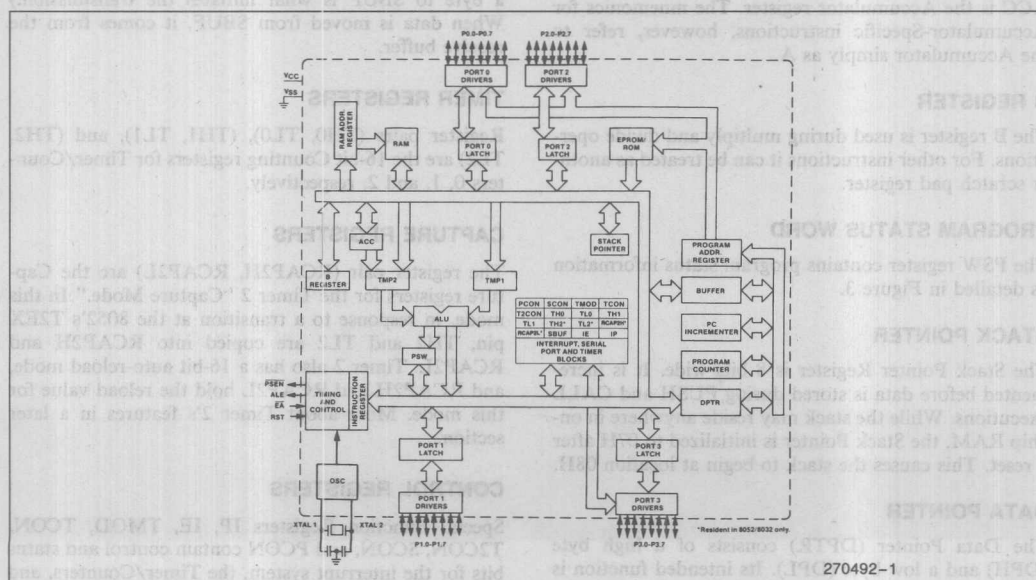


Figure 1. MCS-51 Architectural Block Diagram

8 Bytes							
F8							FF
F0	B						F7
E8							EF
E0	ACC						E7
D8							DF
D0	PSW						D7
C8	(T2CON)	(RCAP2L)	(RCAP2H)	(TL2)	(TH2)		CF
C0							CF
B8	IP						BF
B0	P3						B7
A8	IE						AF
A0	P2						A7
98	SCON	SBUF					9F
90	P1						97
88	TCON	TMOD	TL0	TL1	TH0	TH1	8F
80	P0	SP	DPL	DPH			87
						PCON	

Figure 2. SFR Map. (...) Indicates Resident in 8052s, not in 8051s

Note that not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

The functions of the SFRs are outlined below.

ACCUMULATOR

ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

B REGISTER

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

PROGRAM STATUS WORD

The PSW register contains program status information as detailed in Figure 3.

STACK POINTER

The Stack Pointer Register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

DATA POINTER

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is

to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

PORTS 0 TO 3

P0, P1, P2 and P3 are the SFR latches of Ports 0, 1, 2 and 3, respectively.

SERIAL DATA BUFFER

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

TIMER REGISTERS

Register pairs (TH0, TL0), (TH1, TL1), and (TH2, TL2) are the 16-bit Counting registers for Timer/Counters 0, 1, and 2, respectively.

CAPTURE REGISTERS

The register pair (RCAP2H, RCAP2L) are the Capture registers for the Timer 2 "Capture Mode." In this mode, in response to a transition at the 8052's T2EX pin, TH2 and TL2 are copied into RCAP2H and RCAP2L. Timer 2 also has a 16-bit auto-reload mode, and RCAP2H and RCAP2L hold the reload value for this mode. More about Timer 2's features in a later section.

CONTROL REGISTERS

Special Function Registers IP, IE, TMOD, TCON, T2CON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port. They are described in later sections.

(MSB)				(LSB)			
CY	AC	F0	RS1	RS0	OV	—	P
Symbol	Position	Name and Significance					
CY	PSW.7	Carry flag.					
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)					
F0	PSW.5	Flag 0. (Available to the user for general purposes.)					
RS1	PSW.4	Register bank select control bits 1 & 0. Set/cleared by software to determine working register bank (see Note).					
RS0	PSW.3						
Symbol	Position	Name and Significance					
OV	PSW.2	Overflow flag.					
—	PSW.1	User definable flag.					
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity.					

NOTE:
The contents of (RS1, RS0) enable the working register banks as follows:

(0.0)—Bank 0	(00H–07H)
(0.1)—Bank 1	(08H–0FH)
(1.0)—Bank 2	(10H–17H)
(1.1)—Bank 3	(18H–1FH)

Figure 3. PSW: Program Status Word Register

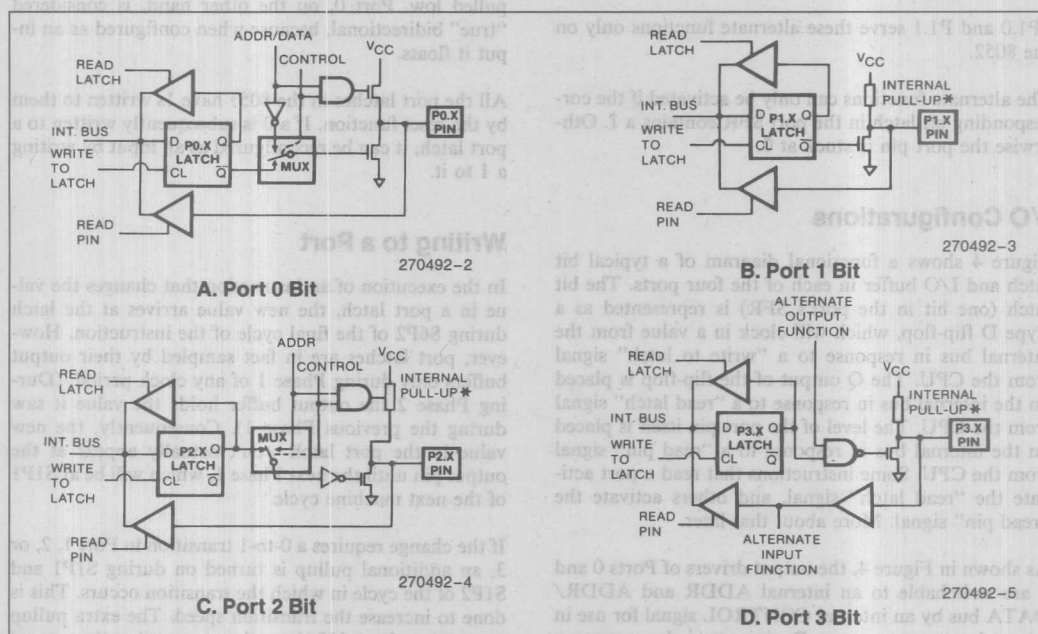


Figure 4. 8051 Port Bit Latches and I/O Buffers

*See Figure 5 for details of the internal pullup.

PORT STRUCTURES AND OPERATION

All four ports in the 8051 are bidirectional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the

external memory address, time-multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise the Port 2 pins continue to emit the P2 SFR content.

All the Port 3 pins, and (in the 8052) two Port 1 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed on the following page.

Port Pin	Alternate Function
*P1.0	T2 (Timer/Counter 2 external input)
*P1.1	T2EX (Timer/Counter 2 Capture/Reload trigger)
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt)
P3.3	INT1 (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	WR (external Data Memory write strobe)
P3.7	RD (external Data Memory read strobe)

*P1.0 and P1.1 serve these alternate functions only on the 8052.

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin is stuck at 0.

I/O Configurations

Figure 4 shows a functional diagram of a typical bit latch and I/O buffer in each of the four ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal. More about that later.

As shown in Figure 4, the output drivers of Ports 0 and 2 are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses, the P2 SFR remains unchanged, but the P0 SFR gets 1s written to it.

Also shown in Figure 4, is that if a P3 bit latch contains a 1, then the output level is controlled by the signal labeled "alternate output function." The actual P3.X pin level is always available to the pin's alternate input function, if any.

Ports 1, 2, and 3 have internal pullups. Port 0 has open drain outputs. Each I/O line can be independently used as an input or an output. (Ports 0 and 2 may not be used as general purpose I/O when being used as the

ADDR/DATA BUS). To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. Then, for Ports 1, 2, and 3, the pin is pulled high by the internal pullup, but can be pulled low by an external source.

Port 0 differs in not having internal pullups. The pullup FET in the P0 output driver (see Figure 4) is used only when the Port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, so the pin floats. In that condition it can be used a high-impedance input.

Because Ports 1, 2, and 3 have fixed internal pullups they are sometimes called "quasi-bidirectional" ports. When configured as inputs they pull high and will source current (IIL, in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because when configured as an input it floats.

All the port latches in the 8051 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

Writing to a Port

In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction. However, port latches are in fact sampled by their output buffers only during Phase 1 of any clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1). Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle.

If the change requires a 0-to-1 transition in Port 1, 2, or 3, an additional pullup is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pullup can source about 100 times the current that the normal pullup can. It should be noted that the internal pullups are field-effect transistors, not linear resistors. The pullup arrangements are shown in Figure 5.

In HMOS versions of the 8051, the fixed part of the pullup is a depletion-mode transistor with the gate wired to the source. This transistor will allow the pin to source about 0.25 mA when shorted to ground. In parallel with the fixed pullup is an enhancement-mode transistor, which is activated during S1 whenever the port bit does a 0-to-1 transition. During this interval, if the port pin is shorted to ground, this extra transistor will allow the pin to source an additional 30 mA.

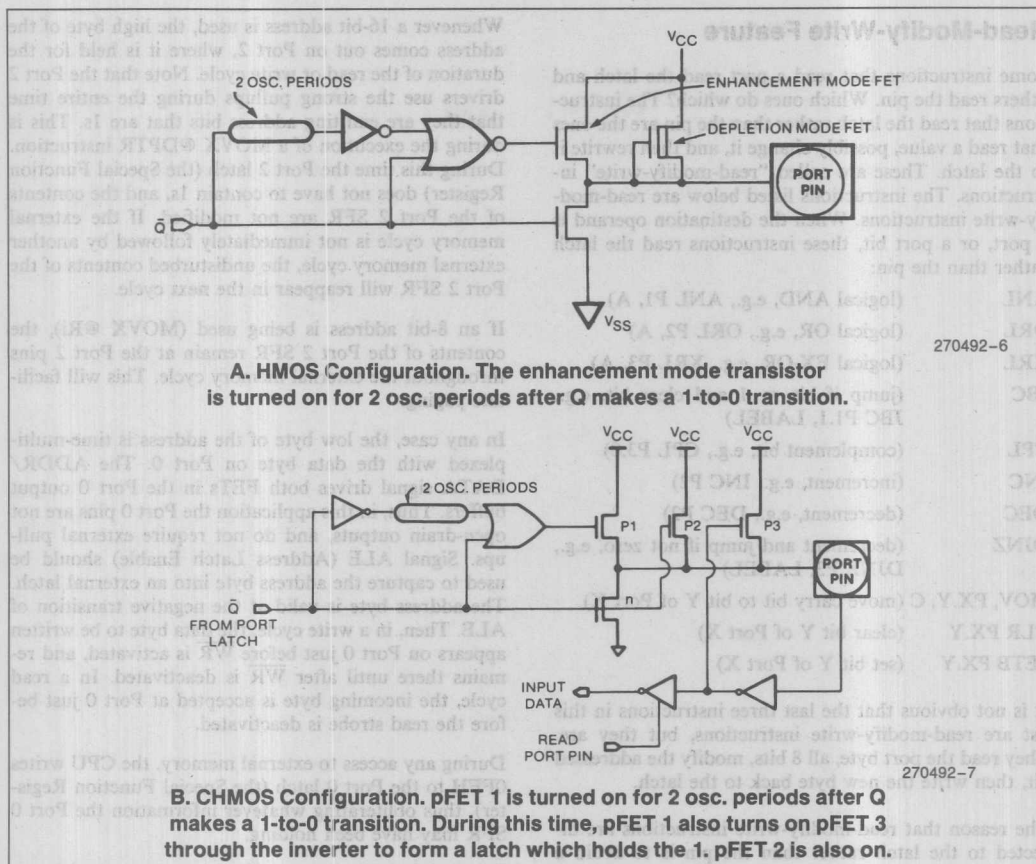


Figure 5. Ports 1 And 3 HMOS And CHMOS Internal Pullup Configurations. Port 2 is Similar Except That It Holds The Strong Pullup On While Emitting 1s That Are Address Bits. (See Text, "Accessing External Memory".)

In the CHMOS versions, the pullup consists of three pFETs. It should be noted that an n-channel FET (nFET) is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

pFET1 in Figure 5 is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. While it's on, it turns on pFET3 (a weak pullup), through the inverter. This inverter and pFET form a latch which hold the 1.

Note that if the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET3, causing the pin to go into a float state. pFET2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about $\frac{1}{10}$ the strength of pFET3. Its function is to restore a 1 to the pin in the event the pin had a 1 and lost it to a glitch.

Port Loading and Interfacing

The output buffers of Ports 1, 2, and 3 can each drive 4 LS TTL inputs. These ports on HMOS versions can be driven in a normal manner by any TTL or NMOS circuit. Both HMOS and CHMOS pins can be driven by open-collector and open-drain outputs, but note that 0-to-1 transitions will not be fast. In the HMOS device, if the pin is driven by an open-collector output, a 0-to-1 transition will have to be driven by the relatively weak depletion mode FET in Figure 5(A). In the CHMOS device, an input 0 turns off pullup pFET3, leaving only the very weak pullup pFET2 to drive the transition.

Port 0 output buffers can each drive 8 LS TTL inputs. They do, however, require external pullups to drive NMOS inputs, except when being used as the ADDRESS/DATA bus.

Read-Modify-Write Feature

Some instructions that read a port read the latch and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions. The instructions listed below are read-modify-write instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

ANL	(logical AND, e.g., ANL P1, A)
ORL	(logical OR, e.g., ORL P2, A)
XRL	(logical EX-OR, e.g., XRL P3, A)
JBC	(jump if bit = 1 and clear bit, e.g., JBC P1.1, LABEL)
CPL	(complement bit, e.g., CPL P3.0)
INC	(increment, e.g., INC P2)
DEC	(decrement, e.g., DEC P2)
DJNZ	(decrement and jump if not zero, e.g., DJNZ P3, LABEL)
MOV, PX.Y, C	(move carry bit to bit Y of Port X)
CLR PX.Y	(clear bit Y of Port X)
SETB PX.Y	(set bit Y of Port X)

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

ACCESSING EXTERNAL MEMORY

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal $\overline{\text{PSEN}}$ (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{\text{RD}}$ or $\overline{\text{WR}}$ (alternate functions of P3.7 and P3.6) to strobe the memory.

Fetches from external Program Memory always use a 16-bit address. Accesses to external Data Memory can use either a 16-bit address (MOVX @DPTR) or an 8-bit address (MOVX @Ri).

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. Note that the Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This is during the execution of a MOVX @DPTR instruction. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. The $\overline{\text{ADDR}}$ / $\overline{\text{DATA}}$ signal drives both FETs in the Port 0 output buffers. Thus, in this application the Port 0 pins are not open-drain outputs, and do not require external pullups. Signal ALE (Address Latch Enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{\text{WR}}$ is activated, and remains there until after $\overline{\text{WR}}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding.

External Program Memory is accessed under two conditions:

- 1) Whenever signal $\overline{\text{EA}}$ is active; or
- 2) Whenever the program counter (PC) contains a number that is larger than 0FFFH (1FFFH for the 8052).

This requires that the ROMless versions have $\overline{\text{EA}}$ wired low to enable the lower 4K (8K for the 8032) program bytes to be fetched from external memory.

When the CPU is executing out of external Program Memory, all 8 bits of Port 2 are dedicated to an output function and may not be used for general purpose I/O. During external program fetches they output the high byte of the PC. During this time the Port 2 drivers use the strong pullups to emit PC bits that are 1s.

TIMER/COUNTERS

The 8051 has two 16-bit Timer/Counter registers: Timer 0 and Timer 1. The 8052 has these two plus one more: Timer 2. All three can be configured to operate either as timers or event counters.

In the "Timer" function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is $1/12$ of the oscillator frequency.

In the "Counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0, T1 or (in the 8052) T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is $1/24$ of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

In addition to the "Timer" or "Counter" selection, Timer 0 and Timer 1 have four operating modes from which to select. Timer 2, in the 8052, has three modes of operation: "Capture," "Auto-Reload" and "baud rate generator."

Timer 0 and Timer 1

These Timer/Counters are present in both the 8051 and the 8052. The "Timer" or "Counter" function is selected by control bits C/T in the Special Function Register TMOD (Figure 6). These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timer/Counters. Mode 3 is different. The four operating modes are described in the following text.

(MSB)				(LSB)			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 0			
GATE Gating operation when set. Timer/Counter "x" is enabled only while "INTx" pin is high and "TRx" control pin is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.				M1 M0 Operating Mode 0 0 MCS-48 Timer "TLx" serves as 5-bit prescaler. 0 1 16-bit Timer/Counter "THx" and "TLx" are cascaded; there is no prescaler. 1 0 8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows. 1 1 (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits. (Timer 1) Timer/Counter 1 stopped.			
C/T Timer or Counter Selector cleared for Timer operation (input from internal system clock). Set for Counter operation (input from "Tx" input pin).							

Figure 6. TMOD: Timer/Counter Mode Control Register

MODE 0

Putting either Timer into Mode 0 makes it look like an 8048 Timer, which is an 8-bit Counter with a divide-by-32 prescaler. Figure 7 shows the Mode 0 operation as it applies to Timer 1.

In this mode, the Timer register is configured as a 13-Bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or INT1 = 1. (Setting GATE = 1 allows the Timer to be controlled by external input INT1, to facilitate pulse width measurements.) TR1 is a control bit in the Special Function Register TCON (Figure 8). GATE is in TMOD.

The 13-Bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the registers.

Mode 0 operation is the same for Timer 0 as for Timer 1. Substitute TR0, TF0 and INT0 for the corresponding Timer 1 signals in Figure 7. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

MODE 1

Mode 1 is the same as Mode 0, except that the Timer register is being run with all 16 bits.

MODE 2

Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload, as shown in Figure 9. Overflow from TL1 not only sets TF1, but also reloads

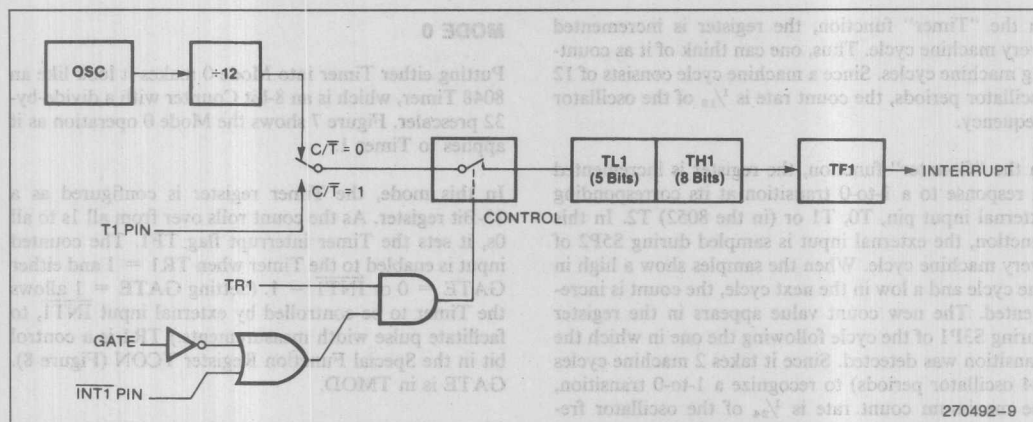


Figure 7. Timer/Counter 1 Mode 0: 13-Bit Counter

(MSB)				(LSB)			
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Symbol	Position	Name and Significance		Symbol	Position	Name and Significance	
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.		IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.	
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.		IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.	
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.		IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.	
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.		IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.	

Figure 8. TCON: Timer/Counter Control Register

TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

Mode 2 operation is the same for Timer/Counter 0.

MODE 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting $TR1 = 0$.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 10. TL0 uses the Timer 0 control bits:

C/\bar{T} , GATE, TR0, $\overline{INT0}$, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the "Timer 1" interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. With Timer 0 in Mode 3, an 8051 can look like it has three Timer/Counters, and an 8052, like it has four. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in fact, in any application not requiring an interrupt.

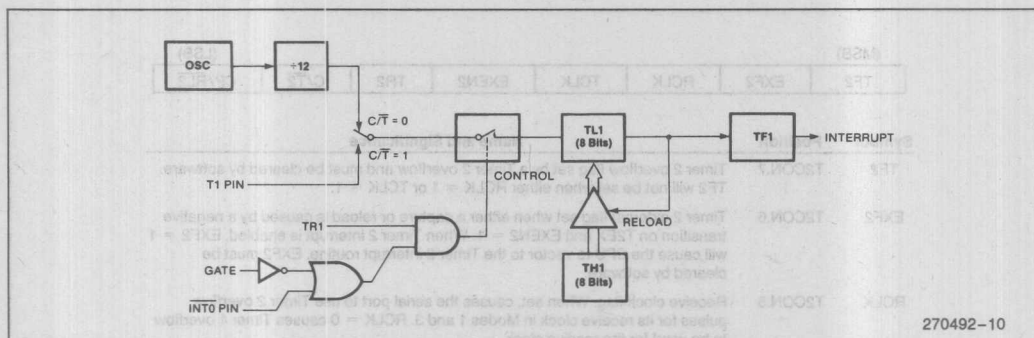


Figure 9. Timer/Counter 1 Mode 2: 8-Bit Auto-Reload

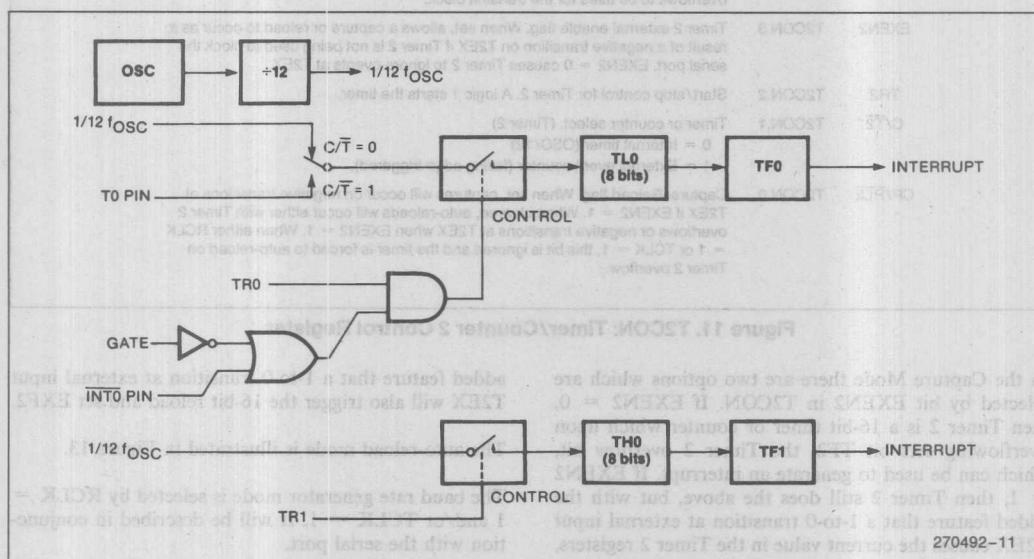


Figure 10. Timer/Counter 0 Mode 3: Two 8-Bit Counters

Timer 2

Timer 2 is a 16-bit Timer/Counter which is present only in the 8052. Like Timers 0 and 1, it can operate either as a timer or as an event counter. This is selected by bit C/T2 in the Special Function Register T2CON (Figure 11). It has three operating modes: "capture," "auto-load" and "baud rate generator," which are selected by bits in T2CON as shown in Table 2.

Table 2. Timer 2 Operating Modes

RCLK + TCLK	CP/RL2	TR2	Mode
0	0	1	16-bit Auto-Reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(off)

(MSB)				(LSB)			
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
Symbol	Position	Name and Significance					
TF2	T2CON.7	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.					
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.					
RCLK	T2CON.5	Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.					
TCLK	T2CON.4	Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.					
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.					
TR2	T2CON.2	Start/stop control for Timer 2. A logic 1 starts the timer.					
C/T2	T2CON.1	Timer or counter select. (Timer 2) 0 = Internal timer (OSC/12) 1 = External event counter (falling edge triggered).					
CP/RL2	T2CON.0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.					

Figure 11. T2CON: Timer/Counter 2 Control Register

In the Capture Mode there are two options which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then Timer 2 is a 16-bit timer or counter which upon overflowing sets bit TF2, the Timer 2 overflow bit, which can be used to generate an interrupt. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. (RCAP2L and RCAP2H are new Special Function Registers in the 8052.) In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2, like TF2, can generate an interrupt.

The Capture Mode is illustrated in Figure 12.

In the auto-reload mode there are again two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then when Timer 2 rolls over it not only sets TF2 but also causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. If EXEN2 = 1, then Timer 2 still does the above, but with the

added feature that a 1-to-0 transition at external input T2EX will also trigger the 16-bit reload and set EXF2.

The auto-reload mode is illustrated in Figure 13.

The baud rate generator mode is selected by RCLK = 1 and/or TCLK = 1. It will be described in conjunction with the serial port.

SERIAL INTERFACE

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost). The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

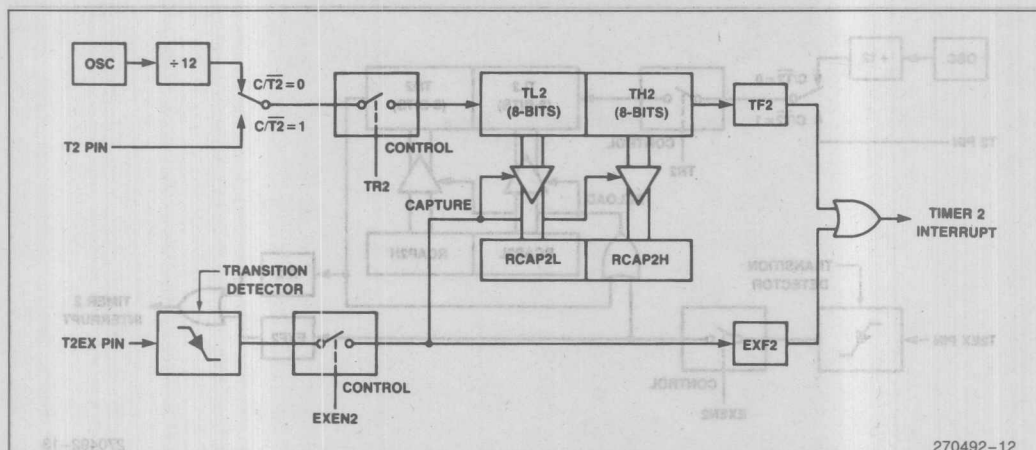


Figure 12. Timer 2 in Capture Mode

The serial port can operate in 4 modes:

Mode 0: Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

Mode 1: 10 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable.

Mode 2: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in Special Function Register SCON, while the stop bit is ignored. The baud rate is programmable to either $1/32$ or $1/64$ the oscillator frequency.

Mode 3: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except the baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications. In these modes, 9 data bits are received. The 9th one goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1. This feature is enabled by setting bit SM2 in SCON. A way to use this feature in multiprocessor systems is as follows.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2s set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

Serial Port Control Register

The serial port control and status register is the Special Function Register SCON, shown in Figure 14. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).

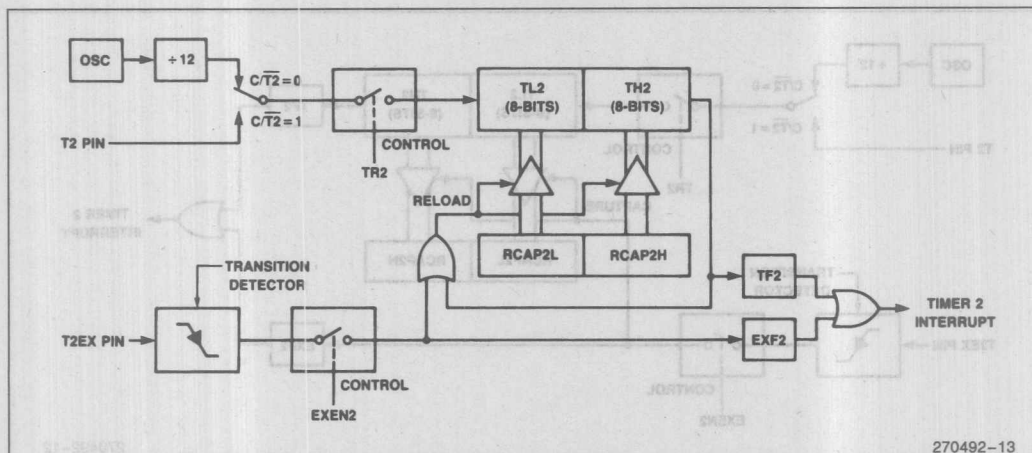


Figure 13. Timer 2 in Auto-Reload Mode

(MSB)					(LSB)			
SM0	SM1	SM2	REN	TB8	RB8	TI	RI	

Where SM0, SM1 specify the serial port mode, as follows:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	$f_{osc}/12$
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	$f_{osc}/64$
				or
				$f_{osc}/32$
1	1	3	9-bit UART variable	

- SM2 enables the multiprocessor communication feature in Modes 2 and 3. In Mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In Mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In Mode 0, SM2 should be 0.
- REN enables serial reception. Set by software to enable reception. Clear by software to disable reception.
- TB8 is the 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
- RB8 in Modes 2 and 3, is the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
- TI is transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
- RI is receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

Figure 14. SCON: Serial Port Control Register

Baud Rates

The baud rate in Mode 0 is fixed:

$$\text{Mode 0 Baud Rate} = \frac{\text{Oscillator Frequency}}{12}$$

The baud rate in Mode 2 depends on the value of bit SMOD in Special Function Register PCON. If SMOD = 0 (which is the value on reset), the baud rate $\frac{1}{64}$ the oscillator frequency. If SMOD = 1, the baud rate is $\frac{1}{32}$ the oscillator frequency.

$$\text{Mode 2 Baud Rate} = \frac{2^{\text{SMOD}}}{64} \times (\text{Oscillator Frequency})$$

In the 8051, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate. In the 8052, these baud rates can be determined by Timer 1, or by Timer 2, or by both (one for transmit and the other for receive).

Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

$$\text{Baud Rate} = \frac{2^{\text{SMOD}}}{32} \times (\text{Timer 1 Overflow Rate})$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In the most typical applications, it is configured for "timer" operation, in the auto-reload

mode (high nibble of TMOD = 0010B). In that case, the baud rate is given by the formula

$$\text{Baud Rate} = \frac{2^{\text{SMOD}}}{32} \times \frac{\text{Oscillator Frequency}}{12 \times [256 - (\text{TH1})]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload.

Figure 15 lists various commonly used baud rates and how they can be obtained from Timer 1.

Baud Rate	fosc	SMOD	Timer 1		
			C/T	Mode	Reload Value
Mode 0 Max: 1 MHz	12 MHz	X	X	X	X
Mode 2 Max: 375K	12 MHz	1	X	X	X
Modes 1, 3: 62.5K	12 MHz	1	0	2	FFH
19.2K	11.059 MHz	1	0	2	FDH
9.6K	11.059 MHz	0	0	2	FDH
4.8K	11.059 MHz	0	0	2	FAH
2.4K	11.059 MHz	0	0	2	F4H
1.2K	11.059 MHz	0	0	2	E8H
137.5K	11.986 MHz	0	0	2	1DH
110K	6 MHz	0	0	2	72H
110K	12 MHz	0	0	1	FEEDH

Figure 15. Timer 1 Generated Commonly Used Baud Rates

Using Timer 2 to Generate Baud Rates

In the 8052, Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Figure

11). Note then the baud rates for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 16.

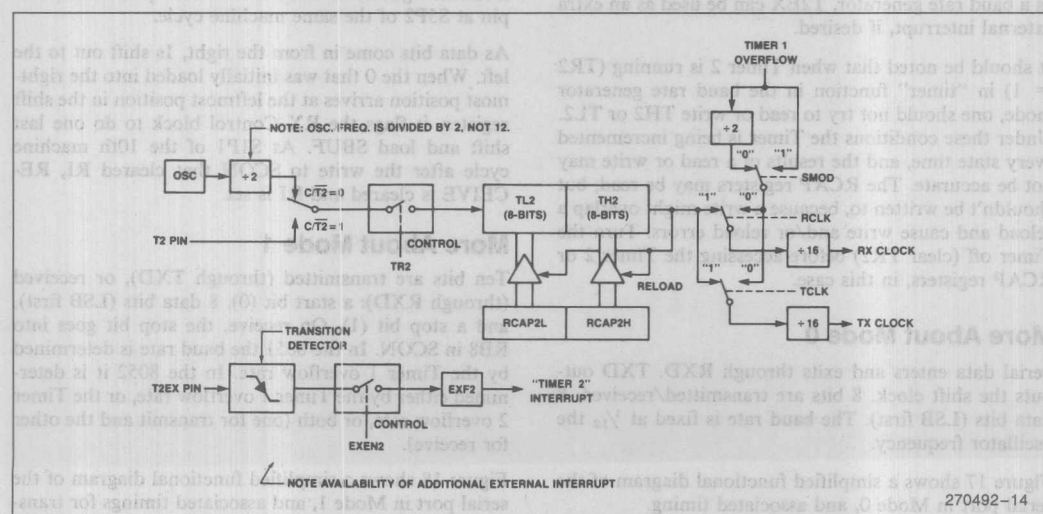


Figure 16. Timer 2 in Baud Rate Generator Mode

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

Now, the baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either "timer" or "counter" operation. In the most typical applications, it is configured for "timer" operation ($C/T2 = 0$). "Timer" operation is a little different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer it would increment every machine cycle (thus at $1/12$ the oscillator frequency). As a baud rate generator, however, it increments every state time (thus at $1/2$ the oscillator frequency). In that case the baud rate is given by the formula

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32x [65536 - (\text{RCAP2H}, \text{RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 16. This Figure is valid only if $\text{RCLK} + \text{TCLK} = 1$ in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running ($\text{TR2} = 1$) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the Timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP registers may be read, but shouldn't be written to, because a write might overlap a reload and cause write and/or reload errors. Turn the Timer off (clear TR2) before accessing the Timer 2 or RCAP registers, in this case.

More About Mode 0

Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at $1/12$ the oscillator frequency.

Figure 17 shows a simplified functional diagram of the serial port in Mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal at S6P2 also loads a 1 into the 9th position of the transmit shift register and tells the TX Control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "write to SBUF," and activation of SEND.

SEND enables the output of the shift register to the alternate output function line of P3.0, and also enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK is low during S3, S4, and S5 of every machine cycle, and high during S6, S1 and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the transmit shift register are shifted to the right one position.

As data bits shift out to the right, zeroes come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control block to do one last shift and then deactivate SEND and set TI. Both of these actions occur at S1P1 of the 10th machine cycle after "write to SBUF."

Reception is initiated by the condition $\text{REN} = 1$ and $\text{R1} = 0$. At S6P2 of the next machine cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates RECEIVE.

RECEIVE enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK makes transitions at S3P1 and S6P1 of every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the P3.0 pin at S5P2 of the same machine cycle.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX Control block to do one last shift and load SBUF. At S1P1 of the 10th machine cycle after the write to SCON that cleared RI, RECEIVE is cleared and RI is set.

More About Mode 1

Ten bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. In the 8051 the baud rate is determined by the Timer 1 overflow rate. In the 8052 it is determined either by the Timer 1 overflow rate, or the Timer 2 overflow rate, or both (one for transmit and the other for receive).

Figure 18 shows a simplified functional diagram of the serial port in Mode 1, and associated timings for transmit receive.

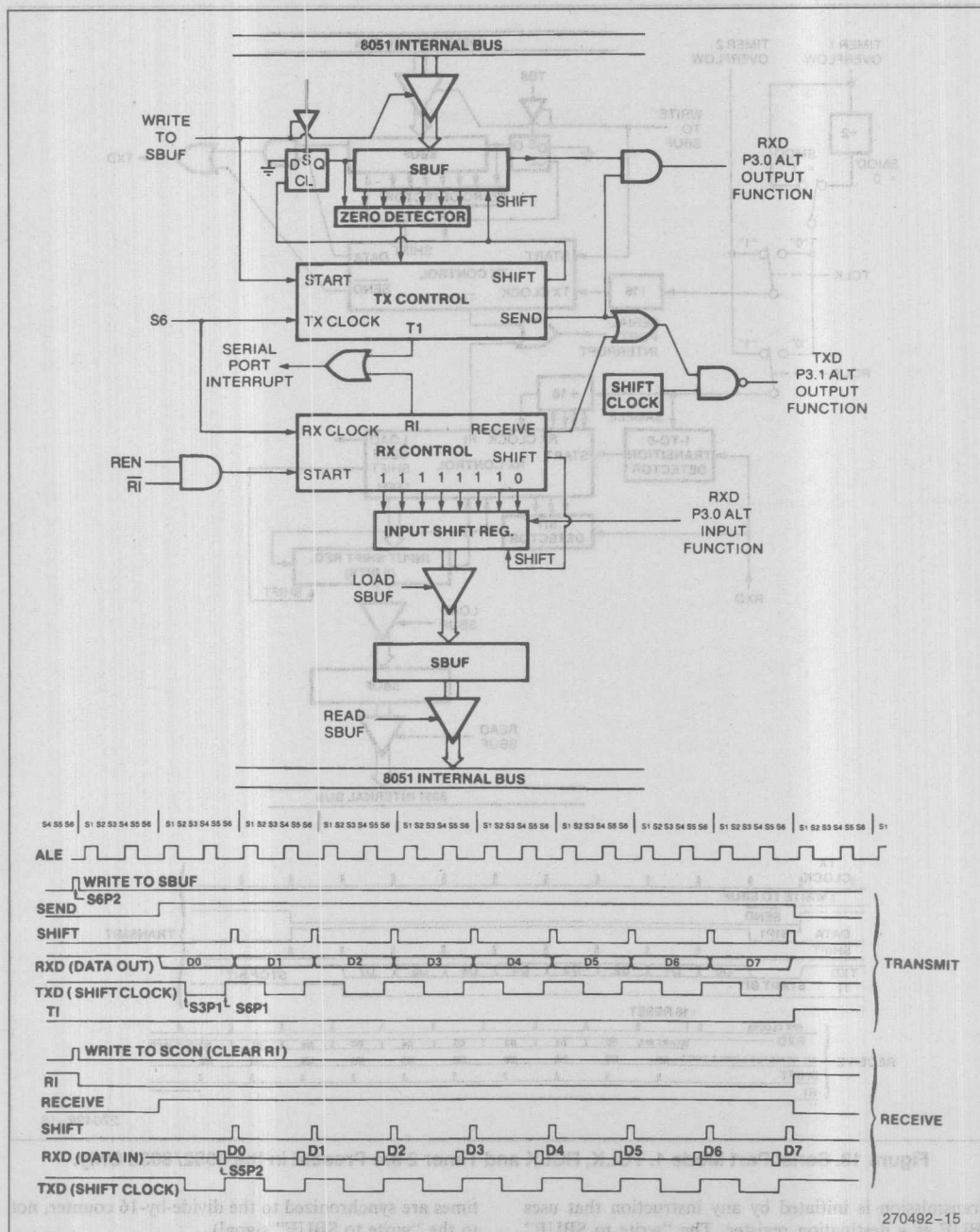


Figure 17. Serial Port Mode 0

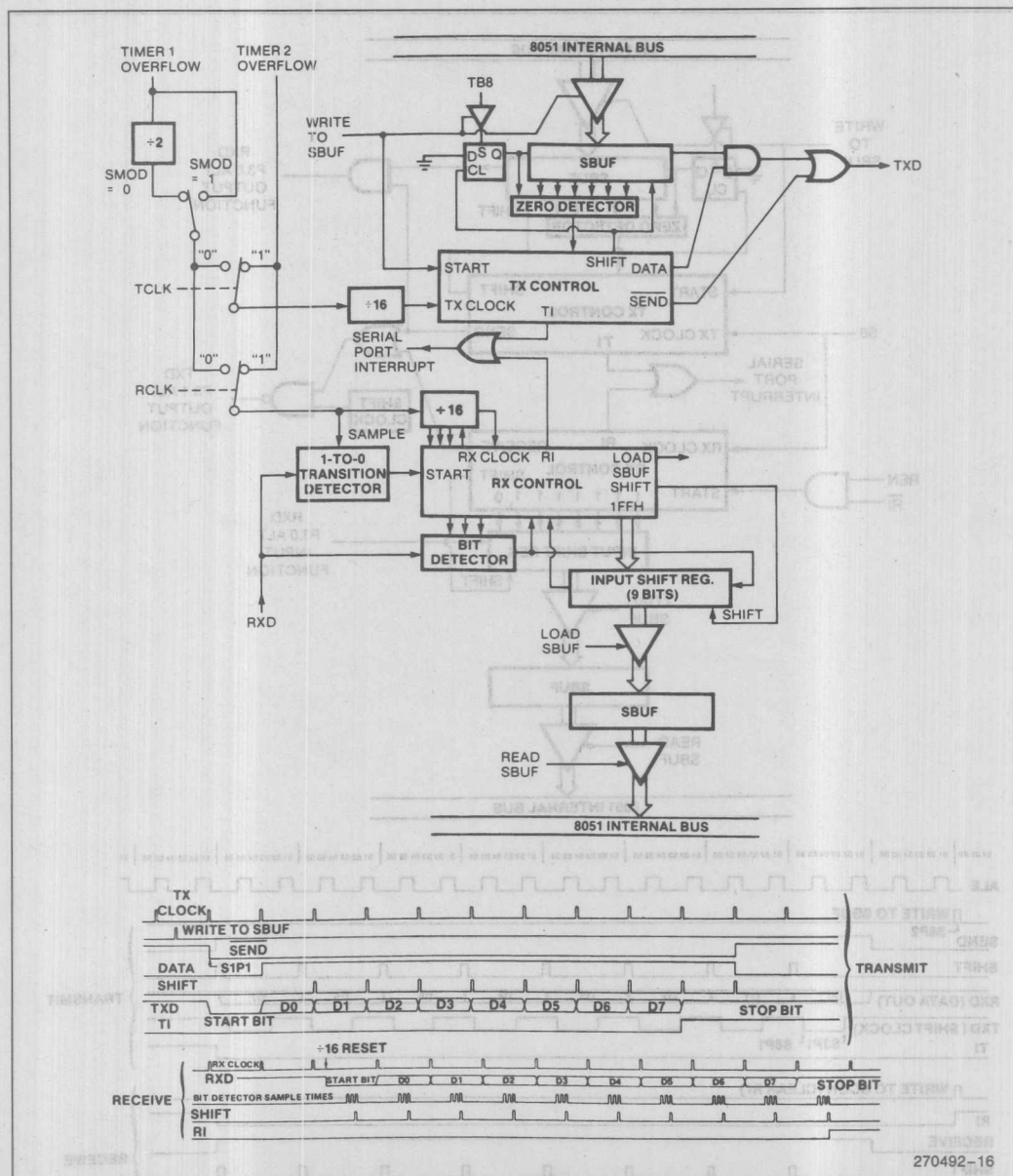


Figure 18. Serial Port Mode 1. TCLK, RCLK and Timer 2 are Present in the 8052/8032 Only.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit

times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal).

The transmission begins with activation of SEND, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeroes are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 10th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register, (which in mode 1 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.

- 1) RI = 0, and
- 2) Either SM2 = 0, or the received stop bit = 1

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition in RXD.

More About Modes 2 and 3

Eleven bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On trans-

mit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to either $\frac{1}{32}$ or $\frac{1}{64}$ the oscillator frequency in Mode 2. Mode 3 may have a variable baud rate generated from either Timer 1 or 2 depending on the state of TCLK and RCLK.

Figures 19 and 20 show a functional diagram of the serial port in Modes 2 and 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeroes are clocked in. Thus, as data bits shift out to the right, zeroes are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 11th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written to the input shift register.

At the 7th, 8th and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.



Figure 19. Serial Port Mode 2

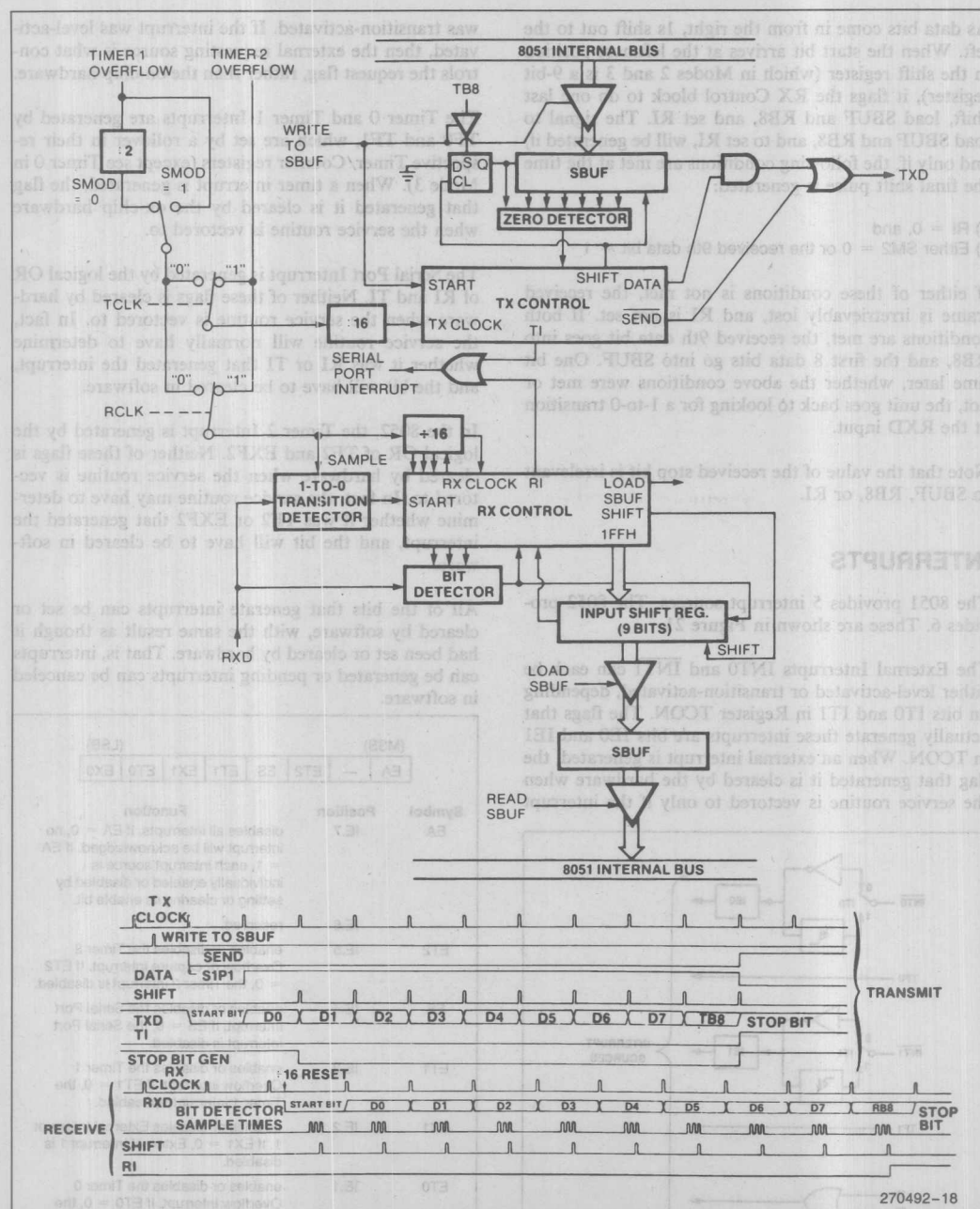


Figure 20. Serial Port Mode 3. TCLK, RCLK, and Timer 2 are Present in the 8052/8032 Only.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in Modes 2 and 3 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

- 1) RI = 0, and
- 2) Either SM2 = 0 or the received 9th data bit = 1

If either of these conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, and the first 8 data bits go into SBUF. One bit time later, whether the above conditions were met or not, the unit goes back to looking for a 1-to-0 transition at the RXD input.

Note that the value of the received stop bit is irrelevant to SBUF, RB8, or RI.

INTERRUPTS

The 8051 provides 5 interrupt sources. The 8052 provides 6. These are shown in Figure 21.

The External Interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt

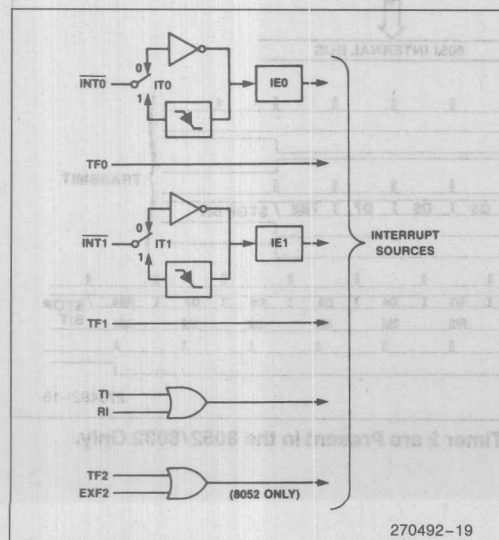


Figure 21. MCS®-51 Interrupt Sources

was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

The Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers (except see Timer 0 in Mode 3). When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The Serial Port Interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

In the 8052, the Timer 2 Interrupt is generated by the logical OR of TF2 and EXF2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

(MSB)				(LSB)			
EA	—	ET2	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved.
ET2	IE.5	enables or disables the Timer 2 Overflow or capture interrupt. If ET2 = 0, the Timer 2 interrupt is disabled.
ES	IE.4	enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

Figure 22. IE: Interrupt Enable Register

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 22). IE contains also a global disable bit, EA, which disables all interrupts at once.

Note in Figure 23 that bit position IE.6 is unimplemented. In the 8051s, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

Priority Level Structure

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP (Figure 23). A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

(MSB)				(LSB)			
—	—	PT2	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
—	IP.7	reserved
—	IP.6	reserved
PT2	IP.5	defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the higher priority level.
PS	IP.4	defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PT0	IP.1	defines the Timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

Figure 23. IP: Interrupt Priority Register

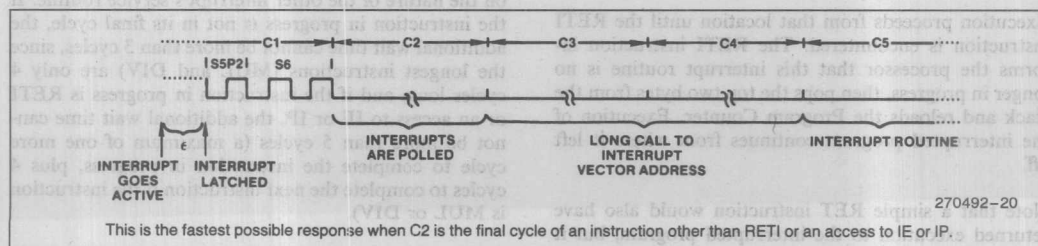


Figure 24. Interrupt Response Timing Diagram

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence, as follows:

Source	Priority Within Level
1. IE0	(highest)
2. TF0	
3. IE1	
4. TF1	
5. RI + TI	
6. TF2 + EXF2	(lowest)

Note that the "priority within level" structure is only used to resolve simultaneous requests of the same priority level.

The IP register contains a number of unimplemented bits. IP.7 and IP.6 are vacant in the 8052s, and in the 8051s these and IP.5 are vacant. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

How Interrupts Are Handled

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be

completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least *one more* instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. Note then that if an interrupt flag is active but not being responded to for one of the above conditions, if the flag is not *still* active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The polling cycle/LCALL sequence is illustrated in Figure 24.

Note that if an interrupt of higher priority level goes active prior to S5P2 of the machine cycle labeled C3 in Figure 24, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the Serial Port or Timer 2 flags. This has to be done in the user's software. It clears an external interrupt flag (IE0 or IE1) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter onto the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to, as shown below.

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress.

External Interrupts

The external sources can be programmed to be level-activated or transition-activated by setting or clearing bit IT1 or IT0 in Register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the INTx pin. If ITx = 1, external interrupt x is edge-triggered. In this mode if successive samples of the INTx pin show a high in one cycle and a low in the next cycle, interrupt request flag IEx in TCON is set. Flag bit IEx then requests the interrupt.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one cycle, and then hold it low for at least one cycle to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If the external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

Response Time

The INT0 and INT1 levels are inverted and latched into IE0 and IE1 at S5P2 of every machine cycle. The values are not actually polled by the circuitry until the next machine cycle. If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine. Figure 24 shows interrupt response timings.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles, since the longest instructions (MUL and DIV) are only 4 cycles long, and if the instruction in progress is RETI or an access to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

SINGLE-STEP OPERATION

The 8051 interrupt structure allows single-step execution with very little software overhead. As previously noted, an interrupt request will not be responded to while an interrupt of equal priority level is still in progress, nor will it be responded to after RETI until at least one other instruction has been executed. Thus, once an interrupt routine has been entered, it cannot be re-entered until at least one instruction of the interrupted program is executed. One way to use this feature for single-stop operation is to program one of the external interrupts (say, INT0) to be level-activated. The service routine for the interrupt will terminate with the following code:

```
JNB P3.2,$ ;Wait Here Till INT0 Goes High
JB P3.2,$ ;Now Wait Here Till it Goes Low
RETI ;Go Back and Execute One Instruction
```

Now if the INT0 pin, which is also the P3.2 pin, is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until INT0 is pulsed (from low to high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re-enter the External Interrupt 0 routine to await the next pulsing of P3.2. One step of the task program is executed each time P3.2 is pulsed.

RESET

The reset input is the RST pin, which is the input to a Schmitt Trigger.

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. The CPU responds by generating an internal reset, with the timing shown in Figure 25.

The external reset signal is asynchronous to the internal clock. The RST pin is sampled during State 5 Phase 2 of every machine cycle. The port pins will maintain their current activities for 19 oscillator periods after a logic 1 has been sampled at the RST pin; that is, for 19 to 31 oscillator periods after the external reset signal has been applied to the RST pin.

While the RST pin is high, ALE and PSEN are weakly pulled high. After RST is pulled low, it will take 1 to 2 machine cycles for ALE and PSEN to start clocking. For this reason, other devices can not be synchronized to the internal timings of the 8051.

The internal reset algorithm writes 0s to all the SFRs except the port latches, the Stack Pointer, and SBUF. The port latches are initialized to FFH, the Stack Pointer to 07H, and SBUF is indeterminate. Table 3 lists the SFRs and their reset values.

The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

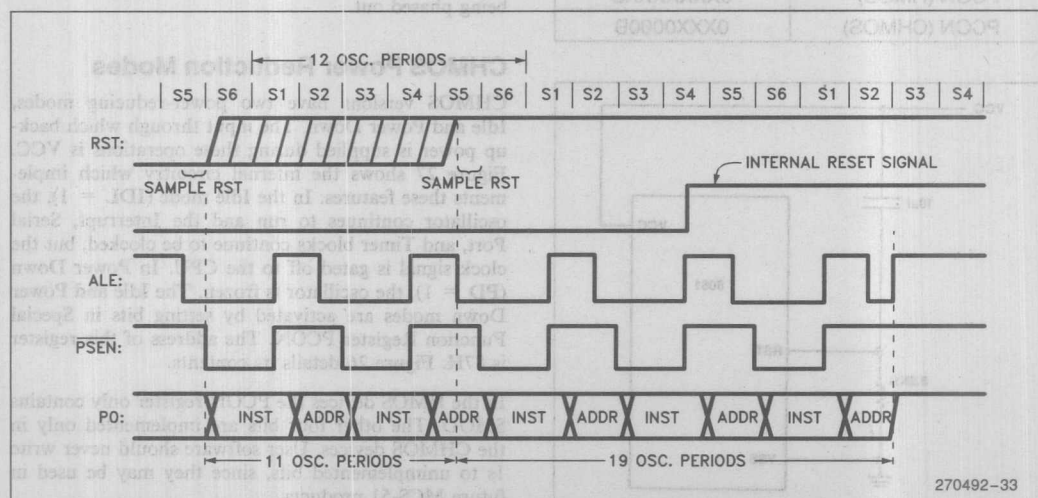


Figure 25. Reset Timing

Table 3. Reset Values of the SFRs

SFR Name	Reset Value
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	FFH
IP (8051)	XXX00000B
IP (8052)	XX000000B
IE (8051)	0XX00000B
IE (8052)	0X000000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
TH2 (8052)	00H
TL2 (8052)	00H
RCAP2H (8052)	00H
RCAP2L (8052)	00H
SCON	00H
SBUF	Indeterminate
PCON (HMOS)	0XXXXXXXB
PCON (CHMOS)	0XXX0000B

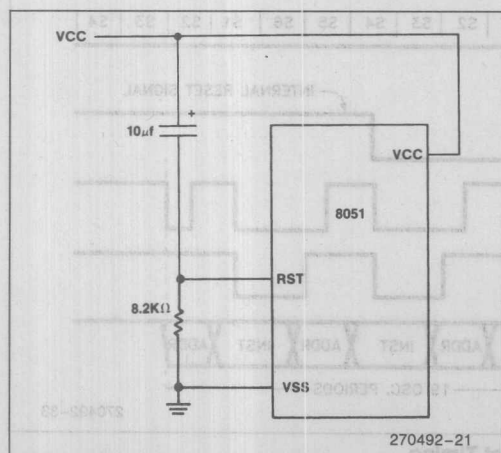


Figure 26. Power on Reset Circuit

POWER-ON RESET

An automatic reset can be obtained when VCC is turned on by connecting the RST pin to VCC through a 10 µf capacitor and to VSS through an 8.2 KΩ resistor, providing the VCC risetime does not exceed a millisecond and the oscillator start-up time does not exceed 10 milliseconds. This power-on reset circuit is shown in Figure 26. The CHMOS devices do not require the 8.2K pull-down resistor, although its presence does no harm.

When power is turned on the circuit holds the RST pin high for an amount of time that depends on the value of the capacitor and the rate at which it charges. To ensure a good reset the RST pin must be high long enough to allow the oscillator time to start up (normally a few msec) plus two machine cycles.

Note that the port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them.

With this circuit, reducing VCC quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited, and will not harm the device.

POWER-SAVING MODES OF OPERATION

For applications where power consumption is critical the CHMOS version provides power reduced modes of operation as a standard feature. The power down mode in HMOS devices is no longer a standard feature and is being phased out.

CHMOS Power Reduction Modes

CHMOS versions have two power-reducing modes, Idle and Power Down. The input through which back-up power is supplied during these operations is VCC. Figure 27 shows the internal circuitry which implements these features. In the Idle mode (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, and Timer blocks continue to be clocked, but the clock signal is gated off to the CPU. In Power Down (PD = 1), the oscillator is frozen. The Idle and Power Down modes are activated by setting bits in Special Function Register PCON. The address of this register is 87H. Figure 26 details its contents.

In the HMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CHMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

IDLE MODE

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle

There are two ways to terminate the Idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into Idle.



Figure 28. PCON: Power Control Register

The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

POWER DOWN MODE

The only exit from Power Down for the 80C51 is a hardware reset. Reset redefines all the SFRs, but does not change the on-chip RAM.

EPROM VERSIONS

10-25

Table 4. EPROM Versions of the 8051 and 8052

Device Name	EPROM Version	EPROM Bytes	Ckt Type	VPP	Time Required to Program Entire Array
8051	(8751)	4K	HMOS	21.0V	4 minutes
8051AH	8751H	4K	HMOS	21.0V	4 minutes
80C51BH	87C51	4K	CHMOS	12.75V	13 seconds
8052AH	8752BH	8K	HMOS	12.75V	26 seconds

The 8752BH and 87C51 use the faster "Quick-Pulse" programming™ algorithm. These devices program at VPP = 12.75V using a series of twenty-five 100 μs PROG pulses per byte programmed. This results in a total programming time of approximately 26 seconds for the 8752BH (8K bytes) and 13 seconds for the 87C51 (4K bytes).

Detailed procedures for programming and verifying each device are given in the data sheets.

EXPOSURE TO LIGHT

It is good practice to cover the EPROM window with an opaque label when the device is in operation. This is not so much to protect the EPROM array from inadvertent erasure, but to protect the RAM and other on-chip logic. Allowing light to impinge on the silicon die while the device is operating can cause logical malfunction.

Program Memory Locks

In some microcontroller applications it is desirable that the Program Memory be secure from software piracy. Intel has responded to this need by implementing a Program Memory locking scheme in some of the MCS-51 devices. While it is impossible for anyone to guarantee absolute security against all levels of technological sophistication, the Program Memory locks in the MCS-51 devices will present a formidable barrier against illegal readout of protected software.

8751H

The 8751H contains a lock bit which, once programmed, denies electrical access by any external means to the on-chip Program Memory. The effect of this lock bit is that while it is programmed the internal Program Memory can not be read out, the device can not be further programmed, and it *can not execute external Program Memory*. Erasing the EPROM array deactivates the lock bit and restores the device's full functionality. It can then be re-programmed.

The procedure for programming the lock bit is detailed in the 8751H data sheet.

87C51 AND 8752BH

The 87C51 and 8752BH contain two Program Memory locking schemes: Encrypted Verify and Lock Bits.

Encrypted Verify: These devices implement a 32-byte EPROM array that can be programmed by the customer, and which can then be used to encrypt the program code bytes during EPROM verification. The EPROM verification procedure is performed as usual, except that each code byte comes out X-NORED with one of the 32 key bytes. The key bytes are gone through in sequence. Therefore, to read the ROM code, one has to know the 32 key bytes in their proper sequence.

Unprogrammed bytes have the value FFH. Therefore, if the Encryption Array is left unprogrammed all the key bytes have the value FFH. Since any code byte X-NORED with FFH leaves the code byte unchanged, leaving the Encryption Array unprogrammed in effect bypasses the encryption feature.

Lock Bits: Also on the chip are two Lock Bits which can be left unprogrammed (U) or programmed (P) to obtain the following features:

Bit 2	Bit 1	Additional Features
U	U	None
U	P	<ul style="list-style-type: none"> Externally fetched code can not access internal Program Memory. Further programming disabled.
P	U	(Reserved for Future definition.)
P	P	<ul style="list-style-type: none"> Externally fetched code can not access internal Program Memory. Further programming disabled. Program verification is disabled.

When Lock Bit 1 is programmed, the logic level at the EA pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of EA be in agreement with the current logic level at that pin in order for the device to function properly.

ONCE Mode in the 87C51

The ONCE ("on-circuit emulation") mode facilitates testing and debugging of systems using the 87C51 without the 87C51 having to be removed from the circuit. The ONCE mode is invoked by:

1. Pull ALE low while the device is in reset and $\overline{\text{PSEN}}$ is high;
2. Hold ALE low as RST is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{\text{PSEN}}$ are weakly pulled high. The oscillator circuit remains active. While the 87C51 is in this mode, an emulator or test CPU can be used to drive the circuit.

Normal operation is restored after a normal reset is applied.

THE ON-CHIP OSCILLATORS

HMOS Versions

The on-chip oscillator circuitry for the HMOS (HMOS-I and HMOS-II) members of the MCS-51 family is a single stage linear inverter (Figure 29), intended for use as a crystal-controlled, positive reactance oscillator (Figure 30). In this application the crystal is operated in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal.

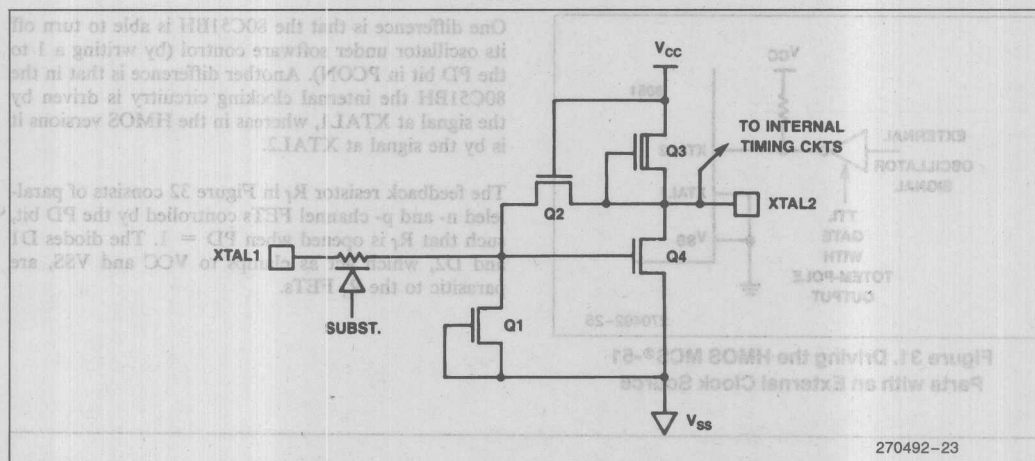


Figure 29. On-Chip Oscillator Circuitry in the HMOS Versions of the MCS®-51 Family

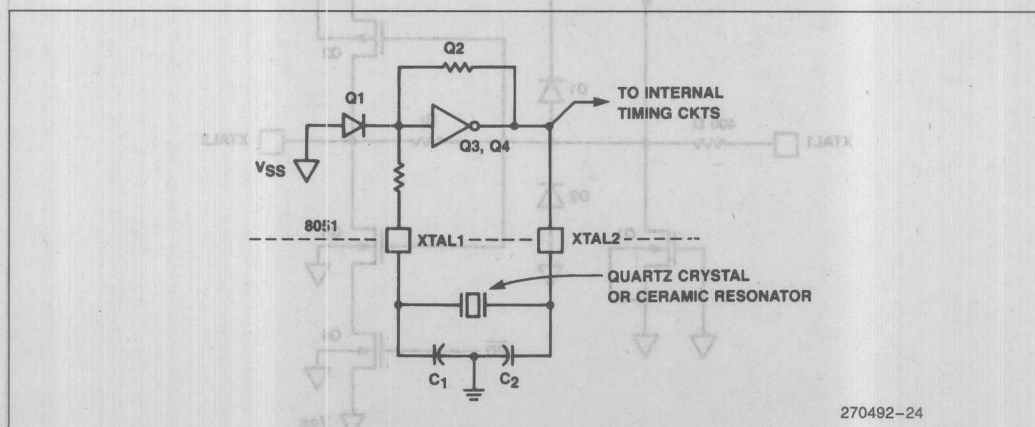


Figure 30. Using the HMOS On-Chip Oscillator

The crystal specifications and capacitance values (C1 and C2 in Figure 30) are not critical. 30 pF can be used in these positions at any frequency with good quality crystals. A ceramic resonator can be used in place of the crystal in cost-sensitive applications. When a ceramic resonator is used, C1 and C2 are normally selected to be of somewhat higher values, typically, 47 pF. The manufacturer of the ceramic resonator should be consulted for recommendations on the values of these capacitors.

A more in-depth discussion of crystal specifications, ceramic resonators, and the selection of values for C1 and C2 can be found in Application Note AP-155, "Oscillators for Microcontrollers," which is included in this manual.

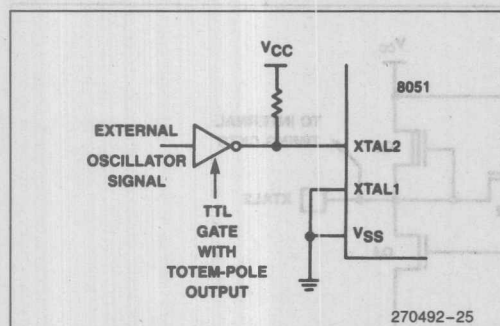


Figure 31. Driving the HMOS MCS[®]-51 Parts with an External Clock Source

To drive the HMOS parts with an external clock source, apply the external clock signal to XTAL2, and ground XTAL1, as shown in Figure 31. A pullup resistor may be used (to increase noise margin), but is optional if V_{OH} of the driving gate exceeds the V_{IH} MIN specification of XTAL2.

CHMOS VERSIONS

The on-chip oscillator circuitry for the 80C51BH, shown in Figure 32, consists of a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator in the same manner as the HMOS parts. However, there are some important differences.

One difference is that the 80C51BH is able to turn off its oscillator under software control (by writing a 1 to the PD bit in PCON). Another difference is that in the 80C51BH the internal clocking circuitry is driven by the signal at XTAL1, whereas in the HMOS versions it is by the signal at XTAL2.

The feedback resistor R_f in Figure 32 consists of paralleled n- and p-channel FETs controlled by the PD bit, such that R_f is opened when PD = 1. The diodes D1 and D2, which act as clamps to VCC and VSS, are parasitic to the R_f FETs.

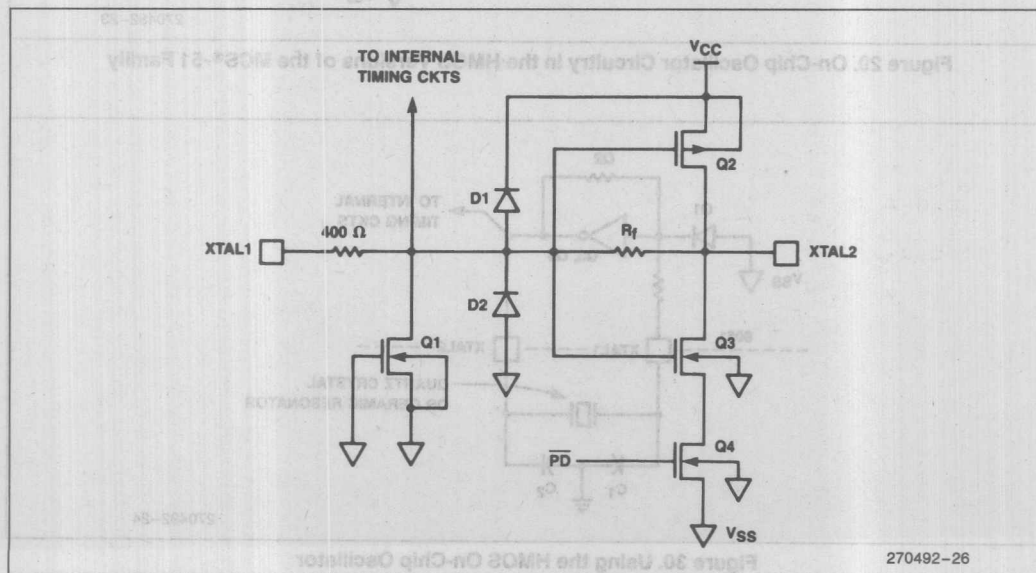


Figure 32. On-Chip Oscillator Circuitry in the CHMOS Versions of the MCS[®]-51 Family

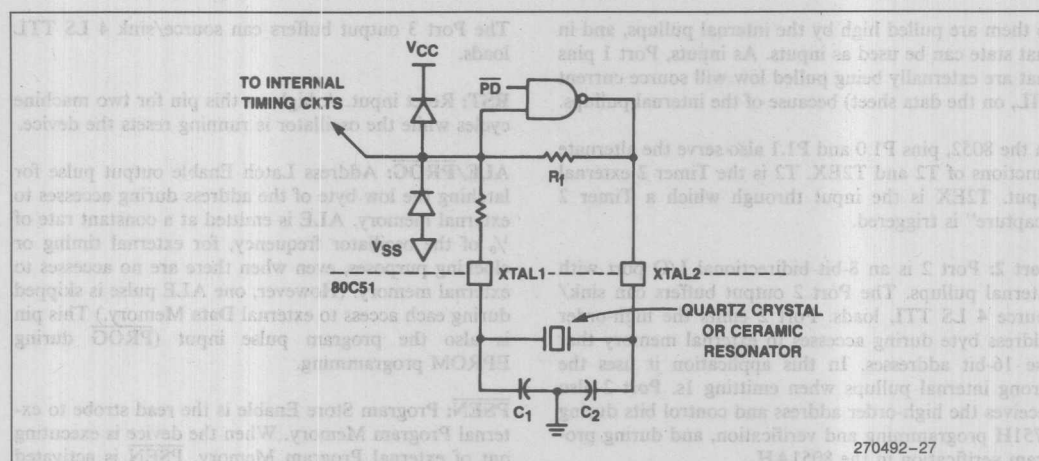


Figure 33. Using the CHMOS On-Chip Oscillator

The oscillator can be used with the same external components as the HMOS versions, as shown in Figure 33. Typically, $C1 = C2 = 30 \text{ pF}$ when the feedback element is a quartz crystal, and $C1 = C2 = 47 \text{ pF}$ when a ceramic resonator is used.

To drive the CHMOS parts with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 float, as shown in Figure 34.

The reason for this change from the way the HMOS part is driven can be seen by comparing Figures 29 and 32. In the HMOS devices the internal timing circuits are driven by the signal at XTAL2. In the CHMOS devices the internal timing circuits are driven by the signal at XTAL1.

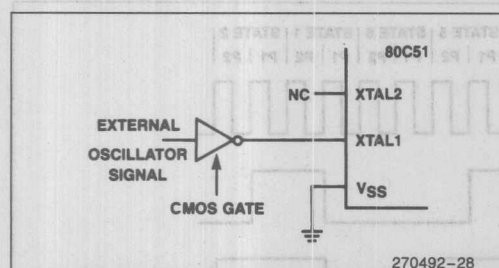


Figure 34. Driving the CHMOS MCS®-51 Parts with an External Clock Source

INTERNAL TIMING

Figures 35 through 38 show when the various strobe and port signals are clocked internally. The figures do not show rise and fall times of the signals, nor do they show propagation delays between the XTAL2 signal and events at other pins.

Rise and fall times are dependent on the external loading that each pin must drive. They are often taken to be something in the neighborhood of 10 nsec, measured between 0.8V and 2.0V.

Propagation delays are different for different pins. For a given pin they vary with pin loading, temperature, VCC, and manufacturing lot. If the XTAL2 waveform is taken as the timing reference, prop delays may vary from 25 to 125 nsec.

The AC Timings section of the data sheets do not reference any timing to the XTAL2 waveform. Rather, they relate the critical edges of control and input signals to each other. The timings published in the data sheets include the effects of propagation delays under the specified test conditions.

MCS®-51 PIN DESCRIPTIONS

VCC: Supply voltage.

VSS: Circuit ground potential.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an open drain output port it can sink 8 LS TTL loads. Port 0 pins that have 1s written to them float, and in that state will function as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external memory. In this application it uses strong internal pullups when emitting 1s. Port 0 also emits code bytes during program verification. In that application, external pullups are required.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL loads. Port 1 pins that have 1s written

to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the internal pullups.

In the 8052, pins P1.0 and P1.1 also serve the alternate functions of T2 and T2EX. T2 is the Timer 2 external input. T2EX is the input through which a Timer 2 "capture" is triggered.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL loads. Port 2 emits the high-order address byte during accesses to external memory that use 16-bit addresses. In this application it uses the strong internal pullups when emitting 1s. Port 2 also receives the high-order address and control bits during 8751H programming and verification, and during program verification in the 8051AH.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. It also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

The Port 3 output buffers can source/sink 4 LS TTL loads.

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/PROG: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. ALE is emitted at a constant rate of $\frac{1}{6}$ of the oscillator frequency, for external timing or clocking purposes, even when there are no accesses to external memory. (However, one ALE pulse is skipped during each access to external Data Memory.) This pin is also the program pulse input (PROG during EPROM programming).

PSEN: Program Store Enable is the read strobe to external Program Memory. When the device is executing out of external Program Memory, PSEN is activated twice each machine cycle (except that two PSEN activations are skipped during accesses to external Data Memory). PSEN is not activated when the device is executing out of internal Program Memory.

EA/VPP: When EA is held high the CPU executes out of internal Program Memory (unless the Program Counter exceeds 0FFFFH in the 8051AH, or 1FFFFH in the 8052). Holding EA low forces the CPU to execute out of external memory regardless of the Program Counter value. In the 8031AH and 8032, EA must be extremely wired low. In the EPROM devices, this pin also receives the programming supply voltage (VPP) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

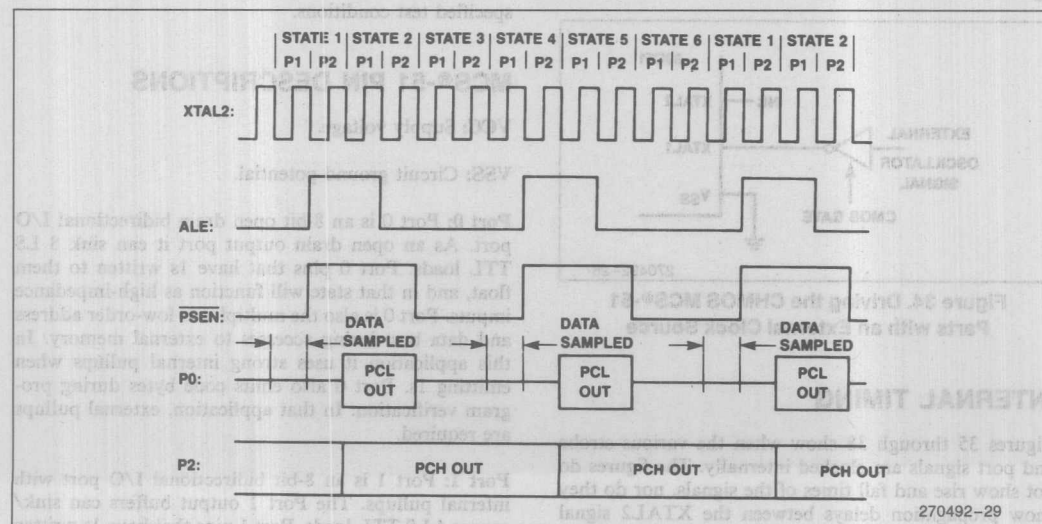


Figure 35. External Program Memory Fetches

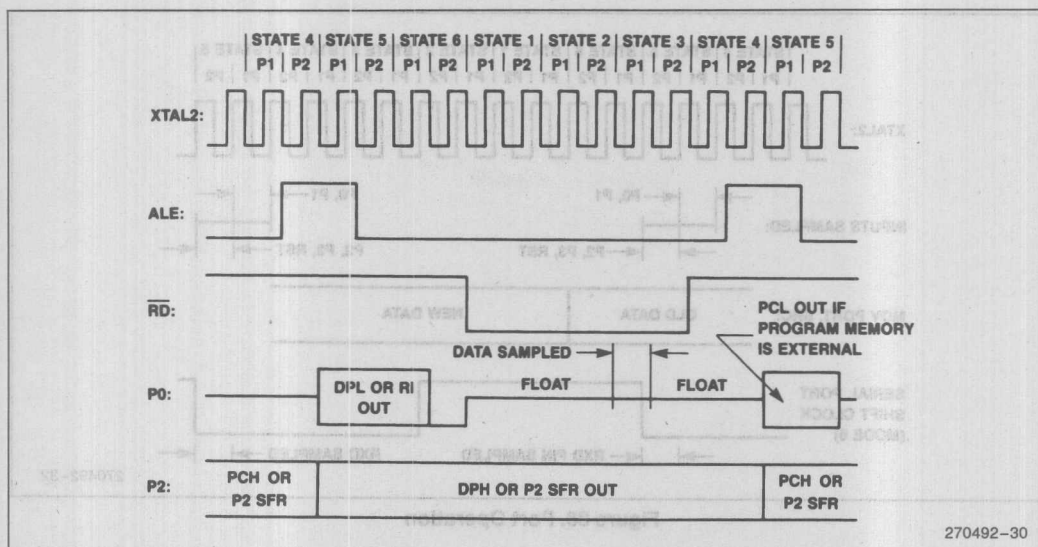


Figure 36. External Data Memory Read Cycle

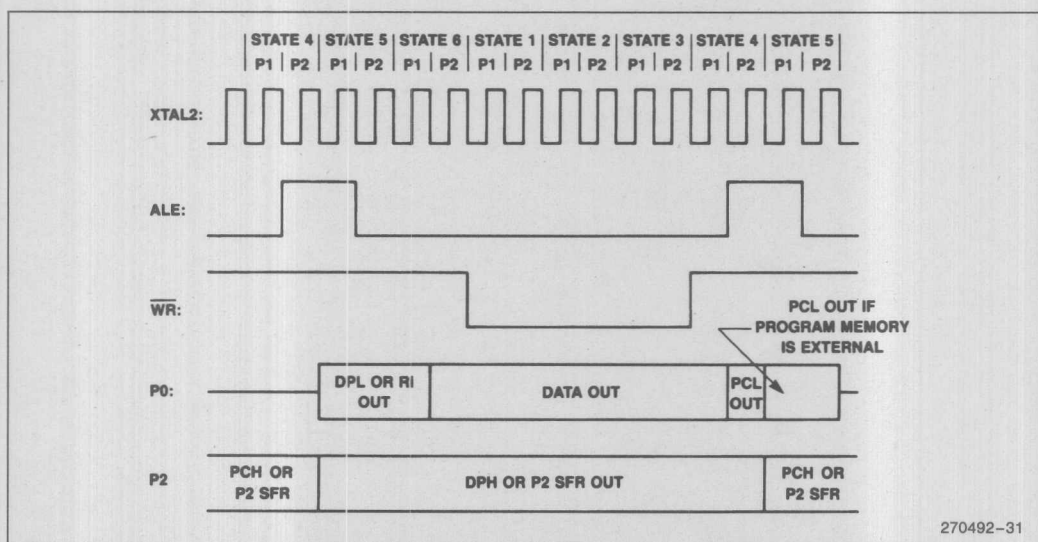


Figure 37. External Data Memory Write Cycle

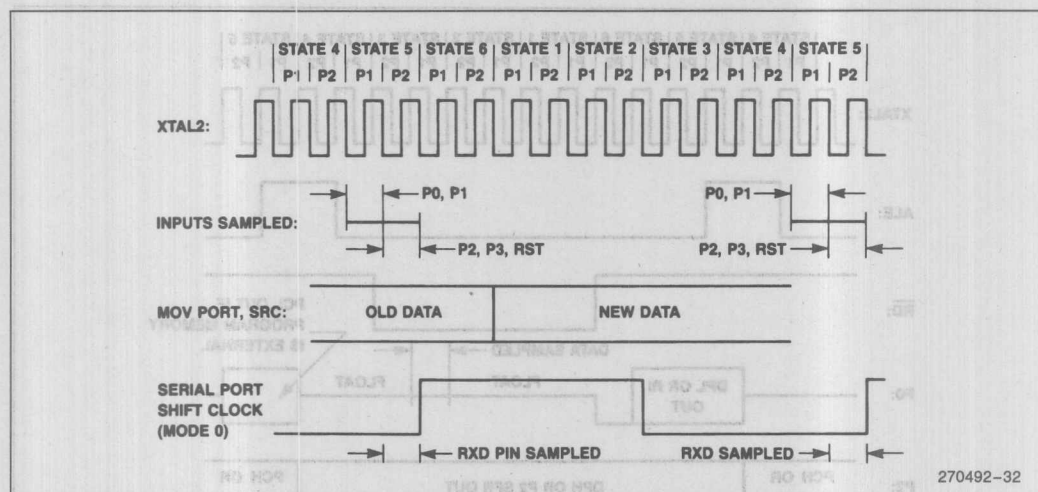


Figure 38. Port Operation

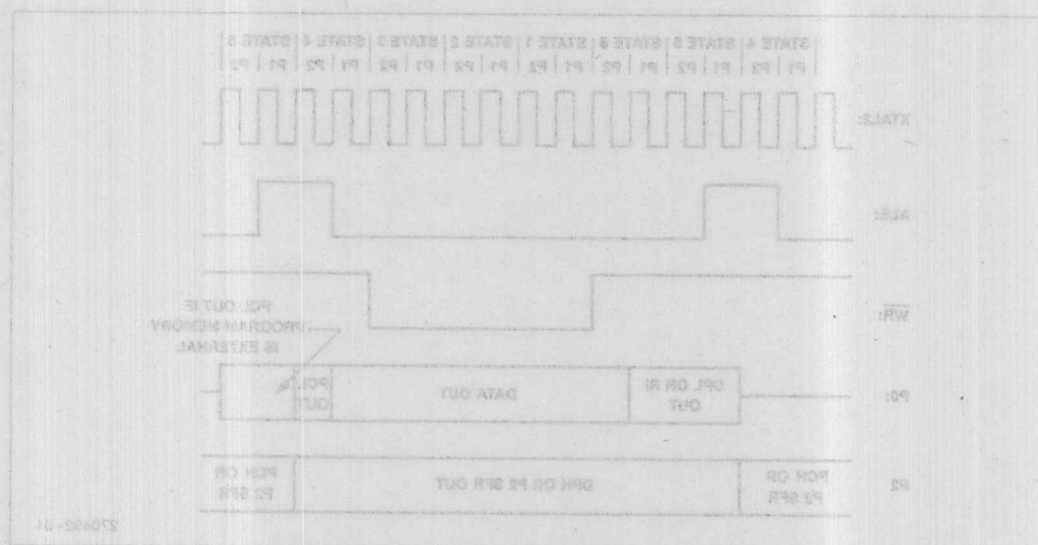


Figure 37. External Data Memory Write Cycle

Hardware Description of the 80C51FA, 83C51FA and 87C51FA

11

11

Hardware Description of the 80C51FA, 83C51FA and 87C51FA

HARDWARE DESCRIPTION OF THE 80C51FA/83C51FA/87C51FA

AUTOMOTIVE

INTRODUCTION

The 80C51FA/83C51FA/87C51FA are 8-bit control-oriented microcontrollers based on the MCS®-51 architecture. The 80C51FA/83C51FA/87C51FA are enhanced versions of the 80C51BH and incorporate many new features. These features include:

■ Programmable Counter Array with

- Compare/Capture
- High Speed Output
- Pulse Width Modulator
- Watchdog Timer

■ Programmable Serial Channel

- Automatic Address Recognition
- Framing Error Detection

The 83C51FA uses the standard 8051 instruction set and is pin for pin compatible with the existing MCS-51 products. However, the numbering system for the 83C51FA is slightly different. The 83C51FA is the factory masked ROM device; the 80C51FA is the ROMless device; and the 87C51FA is the EPROM device. For the remainder of this hardware description the 80C51FA/83C51FA/87C51FA will be referred to as 83C51FA.

It is assumed that the reader is familiar with the 8051 architecture. For more detailed information on the 8051, consult the "Hardware Description of the 8051 and 8052" chapter of this handbook.

OVERVIEW OF THE PCA

The Programmable Timer/Counter Array (PCA) consists of a 16-bit counter, a 16-bit timer, and a 16-bit comparator. Each module has its own mode register, CCAPM, and can be programmed to either run or pause when the CPU is in idle mode. PCA counter share Port 1 pins for hardware interfacing as shown below:

P1.2 ECI	External Count Input to the PCA
P1.3 CE0X	Enhanced Power Down Mode
P1.4 CE1X	16-Bit Up/Down Timer/Counter
P1.5 CE2X	8K Factory Mask ROM/EPROM (83C51FA/87C51FA)
P1.6 CE3X	256 Bytes of On-Chip Data RAM
P1.7 CE4X	7 Interrupt Sources



The Programmable Timer/Counter Array (PCA) consists of a 16-bit counter and five 16-bit compare/capture modules. Each compare/capture module has its own mode register, CCAPMn, which is used to configure the module. The compare/capture modules and the PCA counter share Port 1 pins for hardware interfacing as shown below:

Port Pin	Name	Function
P1.2	ECI	External Count Input to the PCA
P1.3	CEX0	External I/O for Compare/Capture Module 0
P1.4	CEX1	External I/O for Compare/Capture Module 1
P1.5	CEX2	External I/O for Compare/Capture Module 2
P1.6	CEX3	External I/O for Compare/Capture Module 3
P1.7	CEX4	External I/O for Compare/Capture Module 4

The time-base for the PCA is a programmable 16-bit timer/counter. This timer is the only one that can serve the PCA. This timer is started or stopped by setting or clearing bit CR in the Special Function Register CCON, and can be programmed to count any of the following signals (where Fosc is the 83C51FA oscillator frequency):

- Fosc/12
The Counter increments once per machine cycle.
- Fosc/4
With a 16 MHz crystal, the counter increments once every 250 ns.
- Timer 0 overflow
The counter is incremented whenever Timer 0 overflows. This mode allows a programmable input frequency to the PCA.

The counter is incremented when a 1-to-0 transition is detected on the ECI pin. The counter is limited to input frequencies of Fosc/8 in this mode.

The 16-bit PCA timer/counter can also be programmed to either run or pause when the CPU is in Idle mode.

Each of the five 16-bit compare/capture modules can be programmed to do one of the following:

- 16-bit capture, positive edge activated.
- 16-bit capture, negative edge activated.
- 16-bit capture, both positive and negative edge activated.
- 16-bit software timer.
- High-speed output.
- 8-bit Pulse Width Modulator (PWM).

In addition, Compare/Capture module 4 can be used as a Watchdog Timer.

When any of the compare/capture modules are programmed to the capture mode or the 16-bit Timer/High speed output mode, an interrupt can be generated when the module executes its function.

DESCRIPTION OF THE PCA HARDWARE

The time base for the PCA is a 16-bit timer/counter consisting of registers CH and CL (high and low bytes of the count value), controlled by Special Function Register CCON (Figure 1) and a mode register CMOD (Figure 2).

CCON contains bits CF (Counter Flag) which gets set by hardware when the counter rolls over and CR (Counter Run) which is used to turn the counter on and off. It also contains interrupt flags from each of the five PCA modules.

CF	CR	—	CCF4	CCF3	CCF2	CCF1	CCF0
Address = 0D8H			Reset Value = 00X00000B				

Symbol	Position	Function
CF	CCON.7	PCA Counter Overflow flag. Set by hardware when the counter rolls over. CF flags an interrupt if bit ECF in CMOD is set. CF may be set by either hardware or software. It can only be cleared by software.
CR	CCON.6	Counter Run control bit. Set by software to turn the PCA counter on. Clear by software to turn the PCA counter off.
—	CCON.5	Not implemented, reserved for future use.*
CCF4	CCON.4	PCA Module 4 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF3	CCON.3	PCA Module 3 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF2	CCON.2	PCA Module 2 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF1	CCON.1	PCA Module 1 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF0	CCON.0	PCA Module 0 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.

Figure 1. CCON: Counter Control Register

NOTE:

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

CMOD contains the following bits:

CIDL — selects whether the PCA counter continues to run in Idle Mode

WDTE — enables the Watchdog Timer function

CPS1 and CPS0 —select the counter input

ECF — enables CF to generate an interrupt.

CIDL	WDTE	—	—	—	CPS1	CPS0	ECF
Address = 0D9H				Reset Value = 00XXX000B			
Symbol	Position	Function					
CIDL	CMOD.7	Counter Idle control: CIDL = 0 programs PCA Counter to continue functioning during Idle mode. CIDL = 1 programs it to be gated off during Idle.					
WDTE	CMOD.6	Watchdog Timer Enable: WDTE = 0 disables Watchdog Timer function. WDTE = 1 enables it.					
—	CMOD.5	Not implemented, reserved for future use.*					
—	CMOD.4	Not implemented, reserved for future use.*					
—	CMOD.3	Not implemented, reserved for future use.*					
CPS1	CMOD.2	Count Pulse Select bit 1.					
CPS0	CMOD.1	Count Pulse Select bit 0.					
		CPS1	CPS0	PCA Count Pulse Selected			
		0	0	Internal clock, Fosc/12			
		0	1	Internal clock, Fosc/4			
		1	0	Timer 0 overflow			
		1	1	External clock at ECI pin (P1.2) (maximum rate = Fosc/8)			
ECF	CMOD.0	Enable Counter Overflow interrupt: ECF = 1 enables CF bit in CCON to generate an interrupt. ECF = 0 disables that function of CF.					

Figure 2. CMOD: Counter Mode Register

NOTE:

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

Each of the five PCA modules has a Compare/Capture Mode register, CCAPMn, n = 0 through 4 (Figure 3). The following bits in each CCAPMn register define that module's function:

ECOMn — enables that module's comparator function

CAPPn — enables the capture function on positive transitions at the CEXn pin

CAPNn — enables the capture function on negative transitions at the CEXn pin

MATn — enables a comparator match to set the corresponding CCFn flag

TOGn — enables a comparator match to toggle the CEXn pin

PWMn — enables the PWM output at the CEXn pin

ECCFn — enables any Compare/Capture event to flag an interrupt

—	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn
---	-------	-------	-------	------	------	------	-------

Addresses = 0DAH (n=0)
0DBH (n=1)
0DCH (n=2)
0DDH (n=3)
0DEH (n=4)

Reset Value = X0000000B

Symbol	Position	Function
—	CCAPMn.7	Not implemented, reserved for future use.*
ECOMn	CCAPMn.6	ECOMn = 1 enables the comparator function. This bit is automatically cleared by any write to the CCAPnL register, and automatically set by any write to the CCAPnH register. This prevents unintended matches from occurring during writes to the 16-bit Compare/Capture register.
CAPPn	CCAPMn.5	Positive edge capture enable. When CAPPn = 1, a positive transition at the CEXn pin triggers a 16-bit capture from the PCA counter to this module's Compare/Capture register.
CAPNn	CCAPMn.4	Negative edge capture enable. When CAPNn = 1, a negative transition at the CEXn pin triggers a 16-bit capture from the PCA counter to this module's Compare/Capture register.
MATn	CCAPMn.3	When MATn = 1, a match of the PCA Counter with this module's Compare/Capture register causes the CCFn bit in CCON to be set, flagging an interrupt.
TOGn	CCAPMn.2	When TOGn = 1, a match of the PCA Counter with this module's Compare/Capture register causes the CEXn pin to toggle.
PWMn	CCAPMn.1	When PWMn = 1, CEXn is driven high when the low byte of the PCA Counter (CL) matches the low byte of this module's Compare/Capture register (CCAPnL). When CL rolls over to 00H, the CEXn pin is driven low and CCAPnL is updated with the value in CCAPnH. This enables the CEXn pin to be used as a pulse width modulated output. Software varies the pulse width by writing to CCAPnH.
ECCFn	CCAPMn.0	Enables Compare/Capture Flag CCFn in the CCON register to generate an interrupt.

Figure 3. CCAPMn: Compare/Capture Mode Register for PCA Module n

NOTE:

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

There are 6 modes of operation for each of the 5 PCA modules. Shown below are the combinations of bits in the CCAPMn register that are valid and have a

defined function. Invalid combinations will produce undefined results.

ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	Module Function
0	0	0	0	0	0	0	No operation
X	1	0	0	0	0	X	16-bit capture by a positive-edge trigger on CEXn
X	0	1	0	0	0	X	16-bit capture by a negative-edge trigger on CEXn
X	1	1	0	0	0	X	16-bit capture by a transition on CEXn
1	0	0	1	0	0	X	16-bit software timer
1	0	0	1	1	0	X	High Speed Output
1	0	0	0	0	1	0	8-bit PWM

X = Don't Care

16-Bit Timer/Counter

The 5 Compare/Capture modules share a 16-bit timer/counter as their "time base". The timer/counter, shown in Figure 4, can be programmed to increment in 4 different ways. The modes are shown below with the setup values in the CMOD register:

CPS1	CPS0	Method of Incrementing
0	0	Internally clocked at Oscillator Frequency / 12
0	1	Internally clocked at Oscillator Frequency / 4
1	0	Incremented when Timer 0 overflows
1	1	Externally clocked on Pin P1.2/ECI. (Limited to Oscillator Frequency / 8)

16-Bit Capture Mode

Setting CAPPn and/or CAPn puts Compare/Capture module n into Input Capture mode, as shown in Figure 5. The external input pins CEX0 through CEX4, are sampled for a transition. When a valid transition is detected for the current mode of operation (rising edge, falling edge, or either edge), CL is transferred into the CCAPnL register, and CH is transferred into CCAPnH. The resulting value in the Capture Registers, CCAPnL and CCAPnH, reflect the values in CL and CH at the time a transition was detected on the CEXn pin. The event flags an interrupt if bit ECCFn is set.

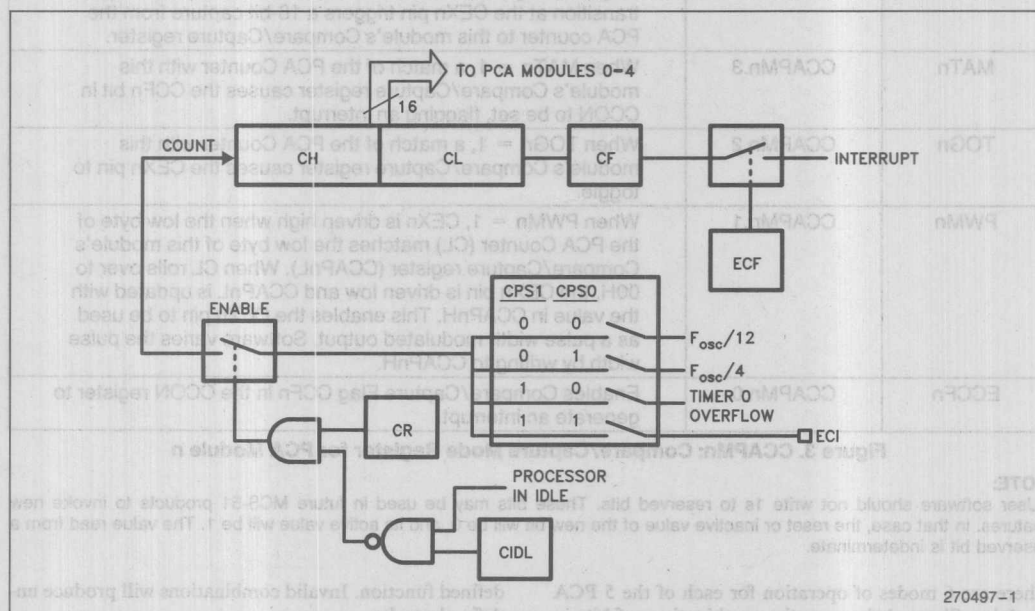


Figure 4. PCA Timer/Counter

Module Function	ECCFn	PWMn	TOGn	MATn	CAPPn	CAPn	ECCFn
16-bit capture by a positive-edge trigger on CEXn	X	0	0	0	0	0	0
16-bit capture by a negative-edge trigger on CEXn	X	0	0	0	0	0	0
16-bit capture by a transition on CEXn	X	0	0	0	0	0	0
16-bit software timer	X	0	0	1	0	0	0
High-Speed Output	X	0	1	1	0	0	0
8-bit PWM	0	1	0	0	0	0	0

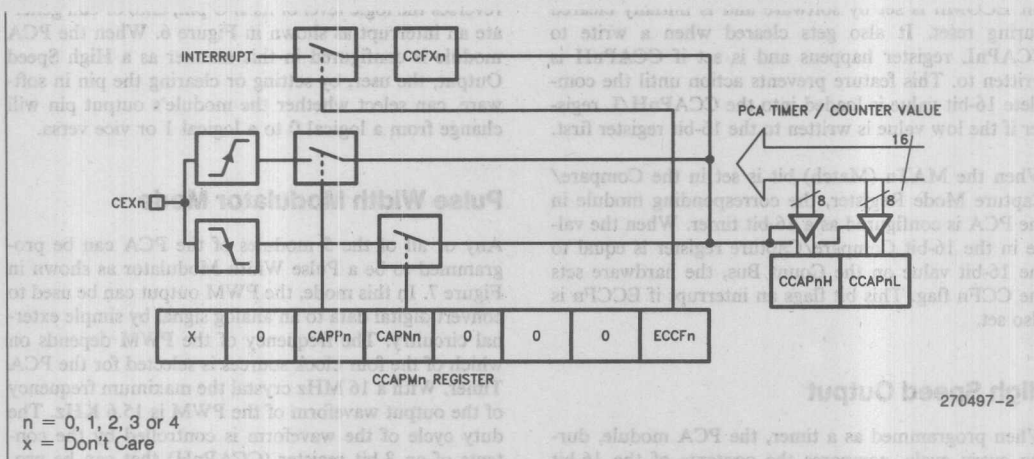


Figure 5. 16-Bit Capture Mode

16 Bit Timer

Setting bit ECOMn in the Compare/Capture Mode Register (CCAPMn) enables the Comparator function as shown in Figure 6. The Comparator compares a 16-bit value stored in the compare/capture register

with the count value of the counter module. When they are equal, a match signal is generated which can set the status bit CCFn in the PCA Control register CCON and/or toggle the corresponding CEXn pin.

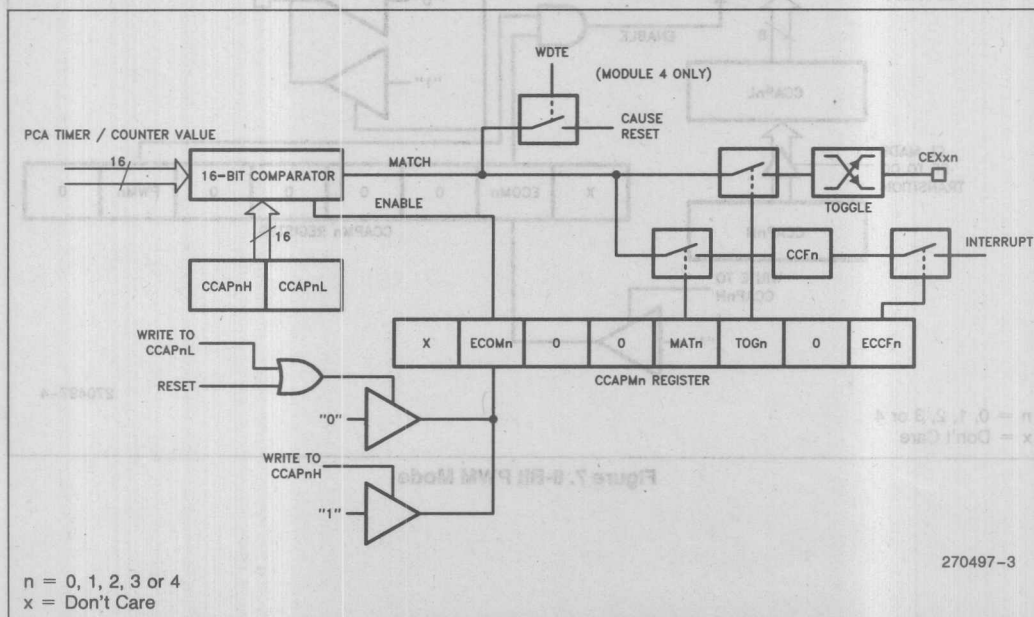


Figure 6. 16-Bit Comparator Mode

Bit ECOMn is set by software and is initially cleared during reset. It also gets cleared when a write to CCAPnL register happens and is set if CCAPnH is written to. This feature prevents action until the complete 16-bit value is loaded into the CCAPnH/L register if the low value is written to the 16-bit register first.

When the MATn (Match) bit is set in the Compare/Capture Mode Register, the corresponding module in the PCA is configured as a 16-bit timer. When the value in the 16-bit Compare/Capture register is equal to the 16-bit value on the Count Bus, the hardware sets the CCFn flag. This bit flags an interrupt if ECCFn is also set.

High Speed Output

When programmed as a timer, the PCA module, during every cycle, compares the contents of the 16-bit timer with the preset value of its Compare registers. When a match occurs, if bit TOGN is set, the module

reverses the logic level of its I/O pin, and/or can generate an interrupt as shown in Figure 6. When the PCA module is configured in this manner as a High Speed Output, the user, by setting or clearing the pin in software, can select whether the module's output pin will change from a logical 0 to a logical 1 or vice versa.

Pulse Width Modulator Mode

Any or all of the 5 modules of the PCA can be programmed to be a Pulse Width Modulator as shown in Figure 7. In this mode, the PWM output can be used to convert digital data to an analog signal by simple external circuitry. The frequency of the PWM depends on which of the four clock sources is selected for the PCA Timer. With a 16 MHz crystal the maximum frequency of the output waveform of the PWM is 15.6 KHz. The duty cycle of the waveform is controlled by the contents of an 8-bit register (CCAPnH) that can be programmed to be any integer from 0 to 255.

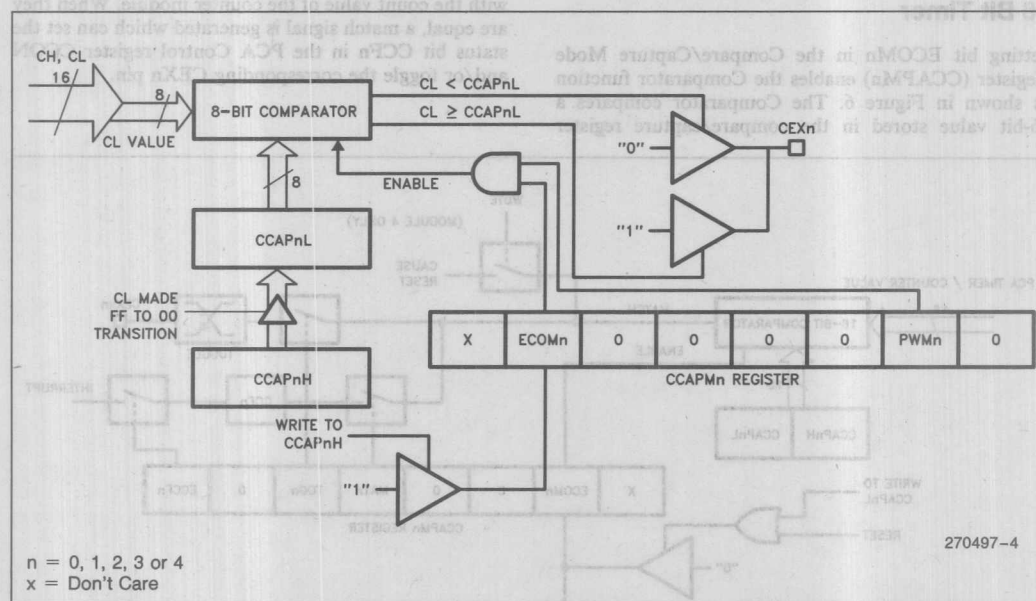


Figure 7. 8-Bit PWM Mode

Watchdog Timer Mode

A Watchdog Timer is a circuit that automatically invokes a reset unless the system being watched sends regular hold-off signals to the Watchdog. These circuits are used in applications that are subject to electrical noise, power glitches, electrostatic discharges, etc., or where high reliability is required with hands-off operation.

In this mode, every time the count in the PCA counter module matches the value stored in compare/capture module 4, an internal reset is generated. The bit that selects this mode is WDTE in the CMOD register. Compare/capture module 4 should be set up to be a 16-bit timer or a High Speed Output in the Watchdog Timer mode.

To hold off the reset, the user's software can:

- Continually reset the PCA 16-bit timer value to a lower value than the reset value in module 4.
- Clear the WDTE bit when a match is about to occur, and then set the WDTE bit just after the match condition (temporarily disabling the feature).
or
- Continually change the CCAP4H and CCAP4L value to one that is "far" from a match value.

Finally, the Watchdog Timer can be used to program a reset by allowing a match to occur.

EXTENDED SERIAL PORT FEATURES

The full duplex serial port of the 83C51FA is the same as the serial port of the 8052 but with two added features: Automatic Address Recognition and Framing Error Detection.

Automatic Address Recognition

Automatic Address Recognition is useful in multi-processor applications in which the CPUs communicate through the serial channel. Using this feature, the 83C51FA's Serial Port refrains from interrupting the CPU unless it receives its own address. Automatic Address Recognition is enabled by setting the SM2 bit in SCON.

Normally the Serial Port would be configured into either of the 9-bit modes (modes 2 and 3). In these modes, if SM2 is set, the Receive Interrupt flag RI is

not activated unless the received byte is an address byte (9th data bit = 1), and the address corresponds to either a Given Address or a Broadcast Address.

The feature works the same way in the 8-bit mode (mode 1) as in the 9-bit modes, except that the stop bit takes the place of the 9th data bit. That is, if SM2 is set, RI is not activated unless the received byte agrees with either the Given or Broadcast address and is terminated by a valid stop bit.

The Given Address is specified by the contents of two new SFRs: SADDR and SADEN. The 80C51FA/83C51FA/87C51FA's individual address is defined in SADDR. SADEN is a mask byte that defines don't-cares in SADDR to form the Given Address. For example,

SADDR = 01010110
SADEN = 11111100

specify the Given Address to be 010101XX.

The Broadcast Address is formed from the logical OR of SADDR and SADEN. Zeros in the logical OR are don't-cares. For example, the values given above for SADDR and SADEN define the broadcast address to be 1111111X.

Automatic Address Recognition allows a host processor to establish communication with an addressed slave, without all the other slave controllers having to respond to the transmission. The addressed slave then clears its SM2 bit to enable reception of data bytes (9th data bit = 0) from the host.

The Given and Broadcast addresses allow each microcontroller to have its own (Given) address and a common (Broadcast) address. A "host" on the serial channel can selectively address single 83C51FA's using the Given Address or all 83C51FA's using the Broadcast Address.

On reset, the SADDR and SADEN registers are initialized to 00H. This defines the Given and Broadcast addresses to be XXXXXXXX (all don't-cares) for backwards compatibility with the MCS-51 family.

Framing Error Detection

Another new feature of the Serial Port is Framing Error Detection. This allows the receiving controller to check the stop bit in modes 1, 2, or 3. A missing stop bit causes a Framing Error bit, FE, to be set. The FE bit can then be checked in software immediately after each reception to detect the lack of a valid stop bit. A missing stop bit can be caused, for example, by noise on the serial lines, or by two CPUs trying to transmit at the same time.

The FE bit, once set, must be cleared in software. A valid stop bit does not cause the FE bit to be cleared.

The FE bit resides in SCON, and has the same bit address as the SM0 bit. A new control bit in the PCON register determines if accesses to the SM0/FE bit address are to SM0 or to FE. The new control bit in PCON is called SMOD0, and resides at PCON.6 (Figure 8). If SMOD0 = 0, then accesses to SCON.7 are to SM0. If SMOD0 = 1, then accesses to SCON.7 are to FE.

REDUCED POWER MODES

Idle Mode

Idle Mode is the same in the 83C51FA as in the 80C51BH. Note that the PCA can be programmed to either pause or continue operations during Idle.

Power Down Mode

The Power Down Mode on the 83C51FA differs from the 80C51BH in one respect: the 83C51FA can exit Power Down with either a hardware Reset or an External Interrupt. (The 80C51BH can only exit Power Down with a hardware Reset.) An exit with an External Interrupt allows not only the on-chip RAM to be saved but also the Special Function Registers.

The External Interrupt, INT0 or INT1, must be enabled and configured as level-sensitive to properly terminate Power Down. Also the interrupt should not be executed before VCC is restored to its normal operating level, and must be held down long enough for the oscillator to restart and stabilize. Once the interrupt is serv-

iced, the next instruction executed after RETI will be the one following the instruction that put the device in Power Down.

Power Off Flag

A Power Off Flag, POF, has been added to the PCON register (Figure 8). This flag is set by hardware when VCC comes up, and can be set or cleared by software. This allows one to distinguish between a "cold start" reset and a "warm start" reset.

A cold start reset is one that is coincident with VCC being turned on to the device after it was turned off. A warm start reset is one that occurs after the device has already been powered up and running. A warm start reset could be generated, for example, by a Watchdog Timer, or as an exit from Power Down Mode.

To use the feature, one checks the POF bit in software immediately after reset. POF = 1 would indicate a cold start. The software then clears POF, and commences its tasks. POF = 0 immediately after reset would indicate a warm start.

VCC must remain above 3 volts for POF to retain a 0.

TIMER 2 AS AN UP/DOWN COUNTER

Timer 2 is a general purpose 16-bit timer/counter which is present in the 8052 and in the 83C51FA. Timer 2 has the same functionality in both of these devices except that in the 8052 Timer 2 can only count up, and in the 83C51FA Timer 2 can be programmed to count up or down. The option to count up or down becomes available when the Timer is configured to its 16-bit auto-reload mode.

SMOD1	SMOD0	X	POF	GF1	GF0	PD	IDL
-------	-------	---	-----	-----	-----	----	-----

Address = 087H

Reset Value = 00XX0000B

- POF** Power Off Flag. Set by hardware on the rising edge of VCC. Set or cleared by software. This flag allows detection of a power failure caused reset. VCC must remain above 3V to retain this bit.
- SMOD0** When set, Read/Write accesses to SCON.7 are to the FE bit. When clear, Read/Write accesses to SCON.7 are to the SM0 bit.
- SMOD1** Same as the SMOD bit in the MSC-51 architecture. The additional bits are defined to be compatible with the 8052 and 80C51BH.

Figure 8. PCON: Power Control Register

The 83C51FA implements a full 256 bytes of on-chip data RAM. As in the 8052, the upper 128 bytes occupy a parallel address space to the Special Function Registers. That means they have the same addresses, but they are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the CPU knows whether the access is to the upper 128 bytes of data RAM or to SFR space by the addressing mode used in the instruction. Instructions that use direct addressing access SFR space. For example,

```
MOV 0A0H, #data
```

accesses the SFR at location 0A0H (which is P2). Instructions that use indirect addressing access the upper 128 bytes of data RAM. For example,

```
MOV @R0, #data
```

where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

PIN DESCRIPTION

The 83C51FA is Pin-for-Pin compatible with the 80C51BH. Port 1 on the 83C51FA has 8052 functionality and additionally serves the PCA as shown below.

Port 1 pins and Alternate Functions

Port Pin	Name	Function
P1.0	T2	External Count Input to Timer 2
P1.1	T2EX	Timer 2 Capture/Reload Trigger
P1.2	ECI	External Count Input to the PCA
P1.3	CEX0	External I/O for Compare/Capture Module 0
P1.4	CEX1	External I/O for Compare/Capture Module 1
P1.5	CEX2	External I/O for Compare/Capture Module 2
P1.6	CEX3	External I/O for Compare/Capture Module 3
P1.7	CEX4	External I/O for Compare/Capture Module 4

P1.0/T2 may be used as an external count input to Timer 2.

P1.1/T2EX can be used to trigger a capture if Timer 2 is in the Capture Mode, or to trigger a reload if Timer 2 is in the Auto-Reload Mode and DCEN is set to 0. T2EX can also control the count direction if Timer 2 is in the Auto-Reload Mode and DCEN is set to 1. Finally, T2EX can be used as an external interrupt if Timer 2 is being used as a baud-rate generator.

P1.2/ECI takes the external signal to the PCA counter. The frequency of the external signal is limited to one eighth of the oscillator frequency or less. The PCA count is incremented every time the ECI pin makes a 1-0 transition.

P1.3 through P1.7/CEXn functions depend on the configuration of their corresponding Compare/Capture modules in the PCA. They can be configured to be a rising edge, falling edge, or an "either edge" trigger input to a Compare/Capture module. They can also be high speed outputs which toggle every time the PCA count matches the value in the corresponding Compare/Capture register. Finally, any of these pins can be configured as an 8-bit Pulse Width Modulated (PWM) output. In the PWM mode, the pin will be in the logical "0" state for a programmable length of time, and will be in the logical "1" state for the remainder of the PWM duty cycle. The PWM duty cycle is variable between 1/256 and 256/256.

Detailed descriptions of the functions of the PCA pins can also be found in section 1.2, PCA feature description.

INTERRUPT STRUCTURE

The 83C51FA provides 7 interrupt sources. Five of them (INT0/ and INT1/, Timer 0 and Timer 1, and the Serial Port) are identical with those provided in the 80C51BH. The 83C51FA also provides a Timer 2 interrupt which is identical with the Timer 2 interrupt in the 8052, and a PCA interrupt which is only found in the 83C51FA. These interrupt sources are shown in Figure 12.

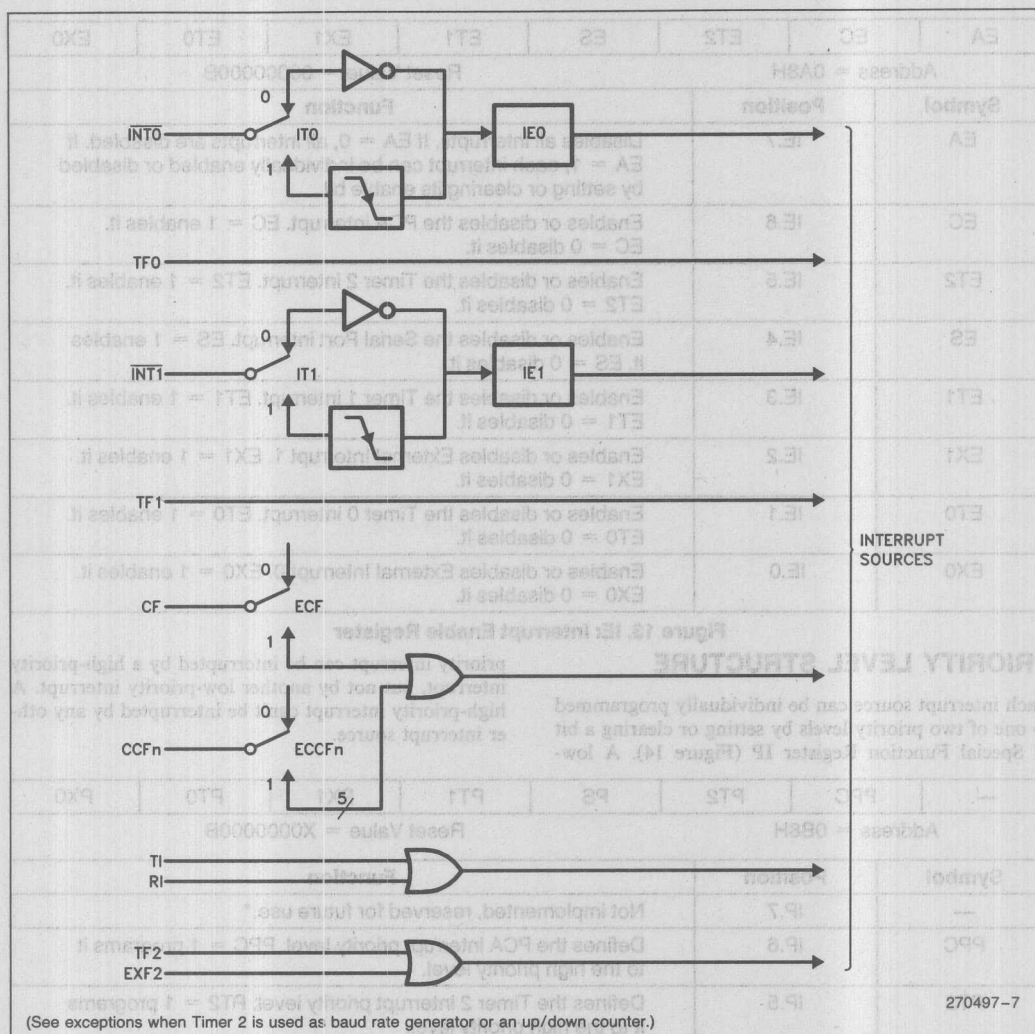


Figure 12. 83C51FA Interrupt Sources

The Timer 2 interrupt is generated by the logical OR of TF2 and EXF2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

The PCA interrupt is generated by the logical OR of CF, CCF0, CCF1, CCF2, CCF3, and CCF4. None of these flags is cleared by hardware when the service routine is vectored to. In fact, normally the service routine will have to determine which bit flagged the interrupt and clear that bit in software.

All of the bits that generate interrupts can be set or cleared in software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled in software.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 13). Note that IE also contains a global disable bit, EA. If EA is set (1), the interrupts are individually enabled or disabled by their corresponding bits in IE. If EA is clear (0), all interrupts are disabled.

EA	EC	ET2	ES	ET1	EX1	ET0	EX0
Address = 0A8H		Reset Value = 0000000B					
Symbol	Position	Function					
EA	IE.7	Disables all interrupts. If EA = 0, all interrupts are disabled. If EA = 1, each interrupt can be individually enabled or disabled by setting or clearing its enable bit.					
EC	IE.6	Enables or disables the PCA interrupt. EC = 1 enables it. EC = 0 disables it.					
ET2	IE.5	Enables or disables the Timer 2 interrupt. ET2 = 1 enables it. ET2 = 0 disables it.					
ES	IE.4	Enables or disables the Serial Port interrupt. ES = 1 enables it. ES = 0 disables it.					
ET1	IE.3	Enables or disables the Timer 1 interrupt. ET1 = 1 enables it. ET1 = 0 disables it.					
EX1	IE.2	Enables or disables External Interrupt 1. EX1 = 1 enables it. EX1 = 0 disables it.					
ET0	IE.1	Enables or disables the Timer 0 interrupt. ET0 = 1 enables it. ET0 = 0 disables it.					
EX0	IE.0	Enables or disables External Interrupt 0. EX0 = 1 enables it. EX0 = 0 disables it.					

Figure 13. IE: Interrupt Enable Register

PRIORITY LEVEL STRUCTURE

Each interrupt source can be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP (Figure 14). A low-

priority interrupt can be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

—	PPC	PT2	PS	PT1	PX1	PT0	PX0
Address = 0B8H			Reset Value = X0000000B				
Symbol	Position	Function					
—	IP.7	Not implemented, reserved for future use.*					
PPC	IP.6	Defines the PCA interrupt priority level. PPC = 1 programs it to the high priority level.					
PT2	IP.5	Defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the high priority level.					
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the high priority level.					
PT1	IP.3	Defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the high priority level.					
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 = 1 programs it to the high priority level.					
PT0	IP.1	Defines the Timer 0 interrupt priority level. PT0 = 1 programs it to the high priority level.					
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the high priority level.					

Figure 14. IP: Interrupt Priority Register

NOTE:

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

If two interrupts of different priority levels are flagged simultaneously, the interrupt request of the higher priority level is serviced. If interrupts of the same priority level are flagged simultaneously, an internal polling sequence determines which interrupt request is serviced. Thus within each priority level there is a second priority structure determined by the following polling sequence:

SOURCE	PRIORITY WITHIN LEVEL
1. IE0	(highest)
2. TF0	
3. IE1	
4. TF1	
5. PCA	
6. RI + TI	
7. TF2 + EXF2	(lowest)

Note that the "priority within level" structure is only used to resolve simultaneous requests of the same priority level.

LOCATION OF INTERRUPT SERVICE ROUTINES

The Interrupt Control System acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate service routine. The hardware-generated LCALL pushes the contents of the Program Counter onto the stack (but it does not save the PSW) and reloads the PC with the starting

location of the interrupt service routine as shown below.

SOURCE STARTING ADDRESS OF SERVICE ROUTINE

IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH
PCA	0033H

Execution proceeds from that location until the RETI instruction is encountered, which terminates the interrupt service routine. Note that the starting addresses of consecutive interrupt service routines are only 8 bytes apart. That means if consecutive interrupts are being used (IE0 and TF0, for example, or TF0 and IE1), and if the first interrupt routine is more than 7 bytes long, then that routine will have to execute a jump out to some other memory location where the service routine can be completed without overlapping the starting address of the next interrupt routine.

Note that, although the polling position of the PCA-generated interrupt is higher than that of the Serial Port, the starting address of the Serial Port interrupt routine is unchanged from the 8051. This is for backwards software compatibility. Similarly, the Timer 2 interrupt starting address is compatible with the 8052. This allows conversion of 8052 (HMOS) designs to the 83C51FA (CHMOS) with no software modification.

SPECIAL FUNCTION REGISTERS

A map of the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

Table 1. Special Function Register Memory Map and Values After Reset

F8		CH	CCAP0H	CCAP1H	CCAP2H	CCAP3H	CCAP4H	FF
		00000000	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	
F0	* B							F7
	00000000							
E8		CL	CCAP0L	CCAP1L	CCAP2L	CCAP3L	CCAP4L	EF
		00000000	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	
E0	* ACC							E7
	00000000							
D8	CCON	CMOD	CCAPM0	CCAPM1	CCAPM2	CCAPM3	CCAPM4	DF
	00X00000	00XXX000	X0000000	X0000000	X0000000	X0000000	X0000000	
D0	* PSW							D7
	00000000							
C8	T2CON	T2MOD	RCAP2L	RCAP2H	TL2	TH2		CF
	00000000	XXXXXXXX0	00000000	00000000	00000000	00000000		
C0								C7
B8	* IP	SADEN						BF
	X0000000	00000000						
B0	* P3							B7
	11111111							
A8	* IE	SADDR						AF
	00000000	00000000						
A0	* P2							A7
	11111111							
98	* SCON	* SBUF						9F
	00000000	XXXXXXXX						
90	* P1							97
	11111111							
88	* TCON	* TMOD	* TL0	* TL1	* TH0	* TH1		8F
	00000000	00000000	00000000	00000000	00000000	00000000		
80	* P0	* SP	* DPL	* DPH			* PCON **	87
	11111111	00000111	00000000	00000000			00XX0000	

* = Found in the 8051 core (See 8051 Hardware Description for explanations of these SFRs).

** = See description of PCON SFR. Bit PCON.4 is not affected by reset.

X = Undefined.

MCS[®]-51 Programmer's Guide and Instruction Set

12

MCS-51 Programmer's Guide and Instruction Set

12



MCS[®]-51 PROGRAMMER'S GUIDE AND INSTRUCTION SET

Automotive

PROGRAM MEMORY

The information presented in this chapter is collected from the previous MCS[®]-51 chapters of this book. The material has been selected and rearranged to form a quick and convenient reference for the programmers of the MCS-51. This guide pertains specifically to the 8051, 8052 and 80C51.

The following list should make it easier to find a subject in this chapter.

Memory Organization

Program Memory	12-2
Data Memory	12-3
Direct and Indirect Address Area	12-5
Special Function Registers	12-7
Contents of SFRs after Power-On	12-8
SFR Memory Map	12-9
Program Status Word (PSW)	12-10
Power Control Register (PCON)	12-10
Interrupts	12-11
Interrupt Enable Register (IE)	12-11
Assigning Priority Level	12-12
Interrupt Priority Register	12-12
Timer/Counter Control Register (TCON)	12-13
Timer/Counter Mode Control Register (TMOD)	12-13
Timer Set-Up	12-14
Timer/Counter 0	12-14
Timer/Counter 1	12-15
Timer/Counter 2 Control Register (T2CON)	12-16
Timer/Counter 2 Set-Up	12-17
Serial Port Control Register	12-18
Serial Port Set-Up	12-18
Generating Baud Rates	12-18
MCS-51 Instruction Set	12-20
Instruction Definitions	12-24

MEMORY ORGANIZATION

PROGRAM MEMORY

The 8051 has separate address spaces for Program Memory and Data Memory. The Program Memory can be up to 64K bytes long. The lower 4K (8K for the 8052) may reside on-chip.

Figure 1 shows a map of the 8051 program memory, and Figure 2 shows a map of the 8052 program memory.

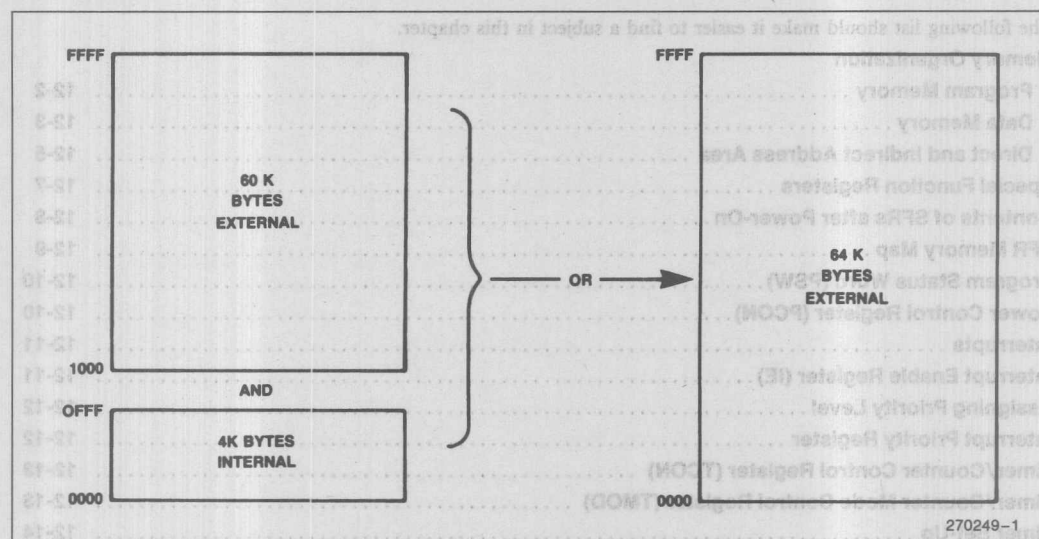


Figure 1. The 8051 Program Memory

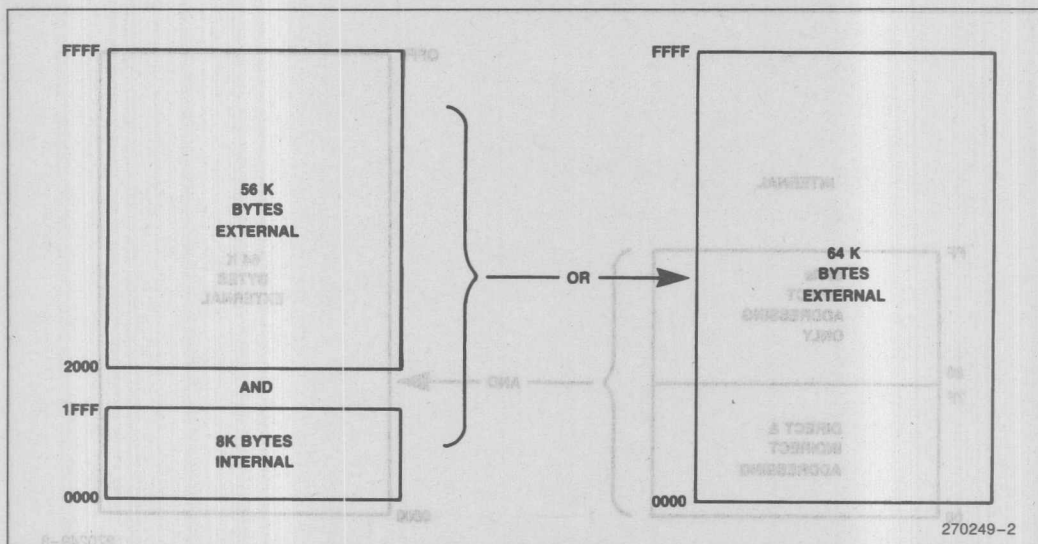


Figure 2. The 8052 Program Memory

Data Memory:

The 8051 can address up to 64K bytes of Data Memory to the chip. The "MOVX" instruction is used to access the external data memory. (Refer to the MCS-51 Instruction Set, in this chapter, for detailed description of instructions).

The 8051 has 128 bytes of on-chip RAM (256 bytes in the 8052) plus a number of Special Function Registers (SFRs). The lower 128 bytes of RAM can be accessed either by direct addressing (MOV data addr) or by indirect addressing (MOV @Ri). Figure 3 shows the 8051 and the 8052 Data Memory organization.

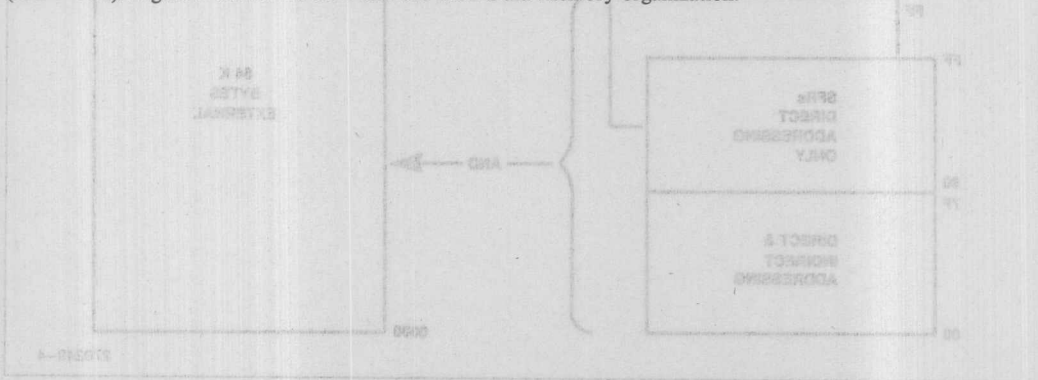


Figure 3. The 8052 Data Memory

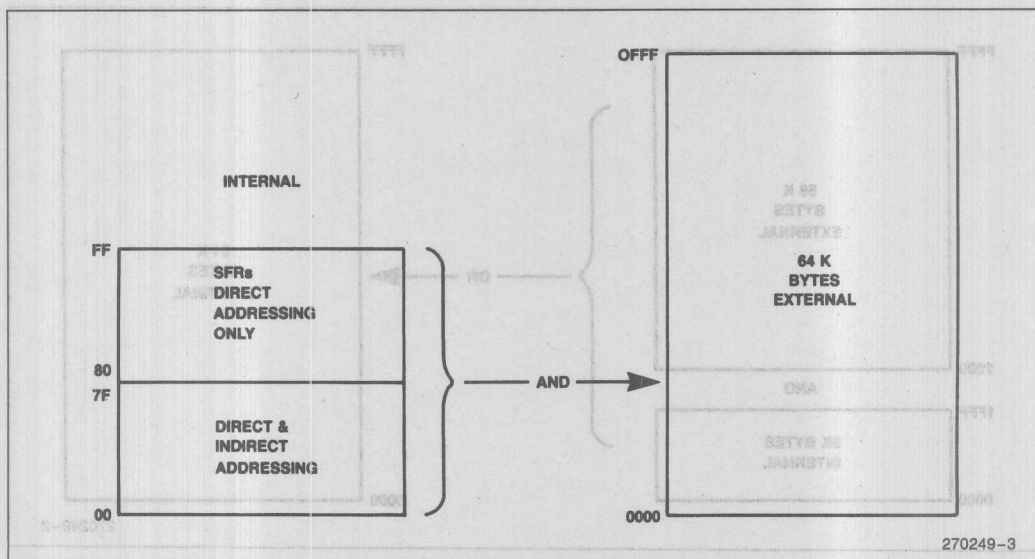


Figure 3a. The 8051 Data Memory

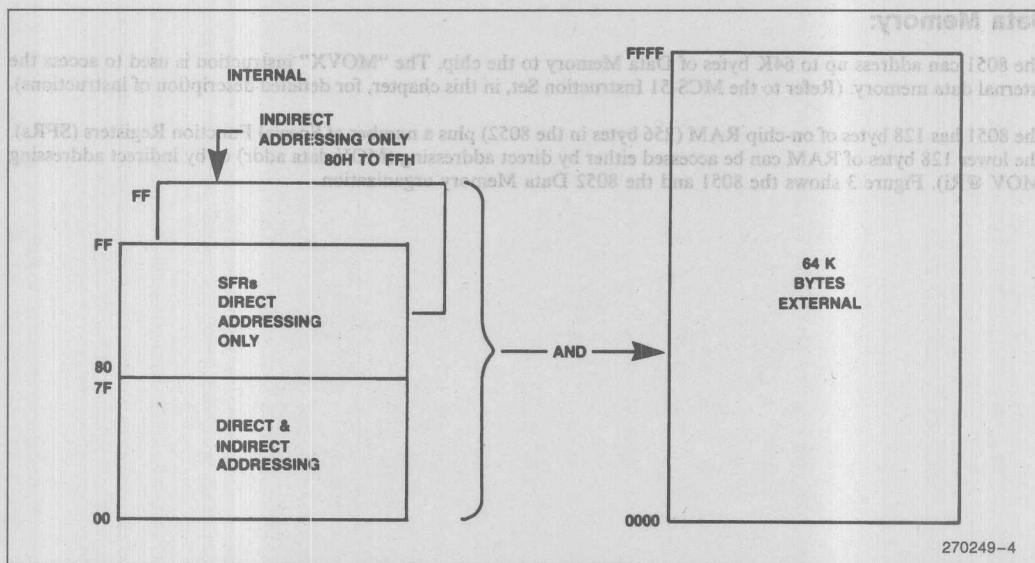


Figure 3b. The 8052 Data Memory

INDIRECT ADDRESS AREA:

Note that in Figure 3b the SFRs and the indirect address RAM have the same addresses (80H–0FFH). Nevertheless, they are two separate areas and are accessed in two different ways.

For example the instruction

```
MOV    80H, #0AAH
```

writes 0AAH to Port 0 which is one of the SFRs and the instruction

```
MOV    R0, #80H
```

```
MOV    @R0, #0BBH
```

writes 0BBH in location 80H of the data RAM. Thus, after execution of both of the above instructions Port 0 will contain 0AAH and location 80 of the RAM will contain 0BBH.

DIRECT AND INDIRECT ADDRESS AREA:

The 128 bytes of RAM which can be accessed by both direct and indirect addressing can be divided into 3 segments as listed below and shown in Figure 4.

1. Register Banks 0-3: Locations 0 through 1FH (32 bytes). ASM-51 and the device after reset default to register bank 0. To use the other register banks the user must select them in the software (refer to the MCS-51 Micro Assembler User's Guide). Each register bank contains 8 one-byte registers, 0 through 7.

Reset initializes the Stack Pointer to location 07H and it is incremented once to start from location 08H which is the first register (R0) of the second register bank. Thus, in order to use more than one register bank, the SP should be initialized to a different location of the RAM where it is not used for data storage (ie, higher part of the RAM).

2. Bit Addressable Area: 16 bytes have been assigned for this segment, 20H-2FH. Each one of the 128 bits of this segment can be directly addressed (0-7FH).

The bits can be referred to in two ways both of which are acceptable by the ASM-51. One way is to refer to their addresses, ie. 0 to 7FH. The other way is with reference to bytes 20H to 2FH. Thus, bits 0–7 can also be referred to as bits 20.0–20.7, and bits 8–FH are the same as 21.0–21.7 and so on.

Each of the 16 bytes in this segment can also be addressed as a byte.

3. Scratch Pad Area: Bytes 30H through 7FH are available to the user as data RAM. However, if the stack pointer has been initialized to this area, enough number of bytes should be left aside to prevent SP data destruction.

Figure 4 shows the different segments of the on-chip RAM.

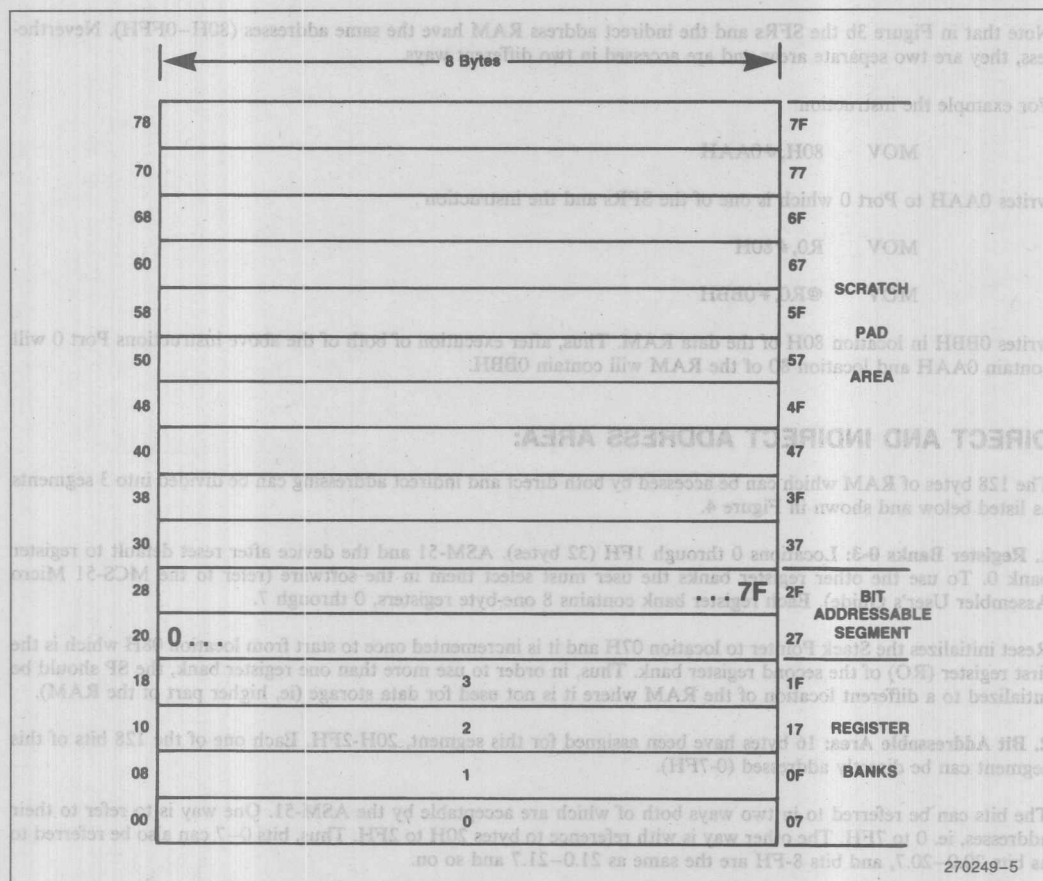


Figure 4. 128 Bytes of RAM Direct and Indirect Addressable

SPECIAL FUNCTION REGISTERS:

Table 1 contains a list of all the SFRs and their addresses.

Comparing Table 1 and Figure 5 shows that all of the SFRs that are byte and bit addressable are located on the first column of the diagram in Figure 5.

Table 1		
Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
*+T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
+TH2	Timer/Counter 2 High Byte	0CDH
+TL2	Timer/Counter 2 Low Byte	0CCH
+RCAP2H	T/C 2 Capture Reg. High Byte	0CBH
+RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

* = Bit addressable

+ = 8052 only

WHAT DO THE SFRs CONTAIN JUST AFTER POWER-ON OR A RESET?

Table 2 lists the contents of each SFR after power-on or a hardware reset.

Table 2. Contents of the SFRs after reset

Register	Value in Binary
*ACC	00000000
*B	00000000
*PSW	00000000
SP	00000111
DPTR	
DPH	00000000
DPL	00000000
*P0	11111111
*P1	11111111
*P2	11111111
*P3	11111111
*IP	8051 XXX00000, 8052 XX000000, 8051 0XX00000, 8052 0X000000
*IE	00000000
TMOD	00000000
*TCON	00000000
*+T2CON	00000000
TH0	00000000
TL0	00000000
TH1	00000000
TL1	00000000
+TH2	00000000
+TL2	00000000
+RCAP2H	00000000
+RCAP2L	00000000
*SCON	00000000
SBUF	Indeterminate
PCON	HMOS 0XXXXXXX CHMOS 0XXX0000

- X = Undefined
 * = Bit Addressable
 + = 8052 only

Those SFRs that have their bits assigned for various functions are listed in this section. A brief description of each bit is provided for quick reference. For more detailed information refer to the Architecture Chapter of this book.

PSW: PROGRAM STATUS WORD. BIT ADDRESSABLE.

CY	AC	F0	RS1	RS0	OV	—	P
CY	PSW.7	Carry Flag.					
AC	PSW.6	Auxiliary Carry Flag.					
F0	PSW.5	Flag 0 available to the user for general purpose.					
RS1	PSW.4	Register Bank selector bit 1 (SEE NOTE 1).					
RS0	PSW.3	Register Bank selector bit 0 (SEE NOTE 1).					
OV	PSW.2	Overflow Flag.					
—	PSW.1	Not implemented, reserved for future use.*					
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bits in the accumulator.					

NOTE:

1. The value presented by RS0 and RS1 selects the corresponding register bank.

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

PCON: POWER CONTROL REGISTER. NOT BIT ADDRESSABLE.

SMOD	—	—	—	GF1	GF0	PD	IDL
------	---	---	---	-----	-----	----	-----

SMOD Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD = 1, the baud rate is doubled when the Serial Port is used in modes 1, 2, or 3.

— Not implemented, reserved for future use.*

— Not implemented, reserved for future use.*

— Not implemented, reserved for future use.*

GF1 General purpose flag bit.

GF0 General purpose flag bit.

PD Power Down bit. Setting this bit activates Power Down operation in the 80C51BH. (Available only in CHMOS).

IDL Idle Mode bit. Setting this bit activates Idle Mode operation in the 80C51BH. (Available only in CHMOS).

If 1s are written to PD and IDL at the same time, PD takes precedence.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

INTERRUPTS:

In order to use any of the interrupts in the MCS-51, the following three steps must be taken.

1. Set the EA (enable all) bit in the IE register to 1.
2. Set the corresponding individual interrupt enable bit in the IE register to 1.
3. Begin the interrupt service routine at the corresponding Vector Address of that interrupt. See Table below.

Interrupt Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI & TI	0023H
TF2 & EXF2	002BH

In addition, for external interrupts, pins $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ (P3.2 and P3.3) must be set to 1, and depending on whether the interrupt is to be level or transition activated, bits IT0 or IT1 in the TCON register may need to be set to 1.

ITx = 0 level activated

ITx = 1 transition activated

IE: INTERRUPT ENABLE REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	—	ET2	ES	ET1	EX1	ET0	EX0
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.					
—	IE.6	Not implemented, reserved for future use.*					
ET2	IE.5	Enable or disable the Timer 2 overflow or capture interrupt (8052 only).					
ES	IE.4	Enable or disable the serial port interrupt.					
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.					
EX1	IE.2	Enable or disable External Interrupt 1.					
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.					
EX0	IE.0	Enable or disable External Interrupt 0.					

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

ASSIGNING HIGHER PRIORITY TO ONE OR MORE INTERRUPTS:

In order to assign higher priority to an interrupt the corresponding bit in the IP register must be set to 1.

Remember that while an interrupt service is in progress, it cannot be interrupted by a lower or same level interrupt.

PRIORITY WITHIN LEVEL:

Priority within level is only to resolve simultaneous requests of the same priority level.

From high to low, interrupt sources are listed below:

IE0
TF0
IE1
TF1
RI or TI
TF2 or EXF2

Interrupt	Vector
TF2 & EXF2	002BH
RI & TI	002BH
TF1	001BH
IE1	001BH
TF0	000BH
IE0	000BH

IP: INTERRUPT PRIORITY REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is 1 the corresponding interrupt has a higher priority.

—	—	PT2	PS	PT1	PX1	PT0	PX0
---	---	-----	----	-----	-----	-----	-----

— IP. 7 Not implemented, reserved for future use.*

— IP. 6 Not implemented, reserved for future use.*

PT2 IP. 5 Defines the Timer 2 interrupt priority level (8052 only).

PS IP. 4 Defines the Serial Port interrupt priority level.

PT1 IP. 3 Defines the Timer 1 interrupt priority level.

PX1 IP. 2 Defines External Interrupt 1 priority level.

PT0 IP. 1 Defines the Timer 0 interrupt priority level.

PX0 IP. 0 Defines the External Interrupt 0 priority level.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

TCON: TIMER/COUNTER CONTROL REGISTER. BIT ADDRESSABLE.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
-----	-----	-----	-----	-----	-----	-----	-----

TF1	TCON. 7	Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	TCON. 6	Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1 ON/OFF.
TF0	TCON. 5	Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.
TR0	TCON. 4	Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.
IE1	TCON. 3	External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT1	TCON. 2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.
IE0	TCON. 1	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.
IT0	TCON. 0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.

GATE	C/T	M1	M0	GATE	C/T	M1	M0
TIMER 1				TIMER 0			

GATE	When TR _x (in TCON) is set and GATE = 1, TIMER/COUNTER _x will run only while INT _x pin is high (hardware control). When GATE = 0, TIMER/COUNTER _x will run only while TR _x = 1 (software control).
C/T	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	Mode selector bit. (NOTE 1)
M0	Mode selector bit. (NOTE 1)

NOTE 1:

M1	M0	Operating Mode
0	0	0 13-bit Timer (MCS-48 compatible)
0	1	1 16-bit Timer/Counter
1	0	2 8-bit Auto-Reload Timer/Counter
1	1	3 (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits.
1	1	3 (Timer 1) Timer/Counter 1 stopped.

TIMER SET-UP

Tables 3 through 6 give some values for TMOD which can be used to set up Timer 0 in different modes.

It is assumed that only one timer is being used at a time. If it is desired to run Timers 0 and 1 simultaneously, in any mode, the value in TMOD for Timer 0 must be ORed with the value shown for Timer 1 (Tables 5 and 6).

For example, if it is desired to run Timer 0 in mode 1 GATE (external control), and Timer 1 in mode 2 COUNTER, then the value that must be loaded into TMOD is 69H (09H from Table 3 ORed with 60H from Table 6).

Moreover, it is assumed that the user, at this point, is not ready to turn the timers on and will do that at a different point in the program by setting bit TRx (in TCON) to 1.

TIMER/COUNTER 0

As a Timer:

Table 3

MODE	TIMER 0 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	08H
1	16-bit Timer	01H	09H
2	8-bit Auto-Reload	02H	0AH
3	two 8-bit Timers	03H	0BH

As a Counter:

Table 4

MODE	COUNTER 0 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	04H	0CH
1	16-bit Timer	05H	0DH
2	8-bit Auto-Reload	06H	0EH
3	one 8-bit Counter	07H	0FH

NOTES:

1. The Timer is turned ON/OFF by setting/clearing bit TR0 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on INT0 (P3.2) when TR0 = 1 (hardware control).

TIMER/COUNTER 1

As a Timer:

Table 5

MODE	TIMER 1 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	80H
1	16-bit Timer	10H	90H
2	8-bit Auto-Reload	20H	A0H
3	does not run	30H	B0H

As a Counter:

Table 6

MODE	COUNTER 1 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	40H	C0H
1	16-bit Timer	50H	D0H
2	8-bit Auto-Reload	60H	E0H
3	not available	—	—

NOTES:

1. The Timer is turned ON/OFF by setting/clearing bit TR1 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on INT1 (P3.3) when TR1 = 1 (hardware control).

T2CON: TIMER/COUNTER 2 CONTROL REGISTER. BIT ADDRESSABLE

8052 Only

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
-----	------	------	------	-------	-----	------	--------

TF2	T2CON. 7	Timer 2 overflow flag set by hardware and cleared by software. TF2 cannot be set when either RCLK = 1 or CLK = 1					
EXF2	T2CON. 6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX, and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.					
RCLK	T2CON. 5	Receive clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its receive clock in modes 1 & 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.					
TCLK	T2CON. 4	Transmit clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its transmit clock in modes 1 & 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.					
EXEN2	T2CON. 3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of negative transition on T2EX if Timer 2 is not being used to clock the Serial Port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.					
TR2	T2CON. 2	Software START/STOP control for Timer 2. A logic 1 starts the Timer.					
C/T2	T2CON. 1	Timer or Counter select. 0 = Internal Timer. 1 = External Event Counter (falling edge triggered).					
CP/RL2	T2CON. 0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, Auto-Reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the Timer is forced to Auto-Reload on Timer 2 overflow.					

TIMER/COUNTER 2 SET-UP

Except for the baud rate generator mode, the values given for T2CON do not include the setting of the TR2 bit. Therefore, bit TR2 must be set, separately, to turn the Timer on.

As a Timer:

Table 7

MODE	T2CON	
	INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
16-bit Auto-Reload	00H	08H
16-bit Capture	01H	09H
BAUD rate generator receive & transmit same baud rate	34H	36H
receive only	24H	26H
transmit only	14H	16H

As a Counter:

Table 8

MODE	TMOD	
	INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
16-bit Auto-Reload	02H	0AH
16-bit Capture	03H	0BH

NOTES:

1. Capture/Reload occurs only on Timer/Counter overflow.
2. Capture/Reload occurs on Timer/Counter overflow and a 1 to 0 transition on T2EX (P1.1) pin except when Timer 2 is used in the baud rate generating mode.

SCON: SERIAL PORT CONTROL REGISTER. BIT ADDRESSABLE.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
-----	-----	-----	-----	-----	-----	----	----

SM0	SCON. 7	Serial Port mode specifier. (NOTE 1).
SM1	SCON. 6	Serial Port mode specifier. (NOTE 1).
SM2	SCON. 5	Enables the multiprocessor communication feature in modes 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0. (See Table 9).
REN	SCON. 4	Set/Cleared by software to Enable/Disable reception.
TB8	SCON. 3	The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.
RB8	SCON. 2	In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.
TI	SCON. 1	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.
RI	SCON. 0	Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.

NOTE 1:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	Fosc./64 OR Fosc./32
1	1	3	9-Bit UART	Variable

SERIAL PORT SET-UP:

Table 9

MODE	SCON	SM2 VARIATION
0	10H	Single Processor Environment (SM2 = 0)
1	50H	
2	90H	
3	D0H	
0	NA	Multiprocessor Environment (SM2 = 1)
1	70H	
2	B0H	
3	F0H	

GENERATING BAUD RATES

Serial Port in Mode 0:

Mode 0 has a fixed baud rate which is 1/12 of the oscillator frequency. To run the serial port in this mode none of the Timer/Counters need to be set up. Only the SCON register needs to be defined.

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{12}$$

Serial Port in Mode 1:

Mode 1 has a variable baud rate. The baud rate can be generated by either Timer 1 or Timer 2 (8052 only).

USING TIMER/COUNTER 1 TO GENERATE BAUD RATES:

For this purpose, Timer 1 is used in mode 2 (Auto-Reload). Refer to Timer Setup section of this chapter.

$$\text{Baud Rate} = \frac{K \times \text{Oscillator Freq.}}{32 \times 12 \times [256 - (\text{TH1})]}$$

If SMOD = 0, then K = 1.

If SMOD = 1, then K = 2. (SMOD is the PCON register).

Most of the time the user knows the baud rate and needs to know the reload value for TH1.

Therefore, the equation to calculate TH1 can be written as:

$$\text{TH1} = 256 - \frac{K \times \text{Osc Freq.}}{384 \times \text{baud rate}}$$

TH1 must be an integer value. Rounding off TH1 to the nearest integer may not produce the desired baud rate. In this case, the user may have to choose another crystal frequency.

Since the PCON register is not bit addressable, one way to set the bit is logical ORing the PCON register. (ie, ORL PCON, #80H). The address of PCON is 87H.

USING TIMER/COUNTER 2 TO GENERATE BAUD RATES:

For this purpose, Timer 2 must be used in the baud rate generating mode. Refer to Timer 2 Setup Table in this chapter. If Timer 2 is being clocked through pin T2 (P1.0) the baud rate is:

$$\text{Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

And if it is being clocked internally the baud rate is:

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{32 \times [65536 - (\text{RCAP2H}, \text{RCAP2L})]}$$

To obtain the reload value for RCAP2H and RCAP2L the above equation can be rewritten as:

$$\text{RCAP2H}, \text{RCAP2L} = 65536 - \frac{\text{Osc Freq}}{32 \times \text{Baud Rate}}$$

SERIAL PORT IN MODE 2:

The baud rate is fixed in this mode and is $\frac{1}{32}$ or $\frac{1}{64}$ of the oscillator frequency depending on the value of the SMOD bit in the PCON register.

In this mode none of the Timers are used and the clock comes from the internal phase 2 clock.

SMOD = 1, Baud Rate = $\frac{1}{32}$ Osc Freq.

SMOD = 0, Baud Rate = $\frac{1}{64}$ Osc Freq.

To set the SMOD bit: ORL PCON, #80H. The address of PCON is 87H.

SERIAL PORT IN MODE 3:

The baud rate in mode 3 is variable and sets up exactly the same as in mode 1.

MCS®-51 INSTRUCTION SET

Table 10. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.							
Instructions that Affect Flag Settings ⁽¹⁾							
Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						
(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.							
Note on instruction set and addressing modes:							
Rn	— Register R7–R0 of the currently selected Register Bank.						
direct	— 8-bit internal data location's address. This could be an Internal Data RAM location (0–127) or a SFR [i.e., I/O port, control register, status register, etc. (128–255)].						
@Ri	— 8-bit internal data RAM location (0–255) addressed indirectly through register R1 or R0.						
#data	— 8-bit constant included in instruction.						
#data 16	— 16-bit constant included in instruction.						
addr 16	— 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.						
addr 11	— 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.						
rel	— Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is –128 to +127 bytes relative to first byte of the following instruction.						
bit	— Direct Addressed bit in Internal Data RAM or Special Function Register.						
*	— New operation not provided by 8048AH/8049AH.						

Mnemonic	Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS			
ADD A,Rn	Add register to Accumulator	1	12
ADD A,direct	Add direct byte to Accumulator	2	12
ADD A,@Ri	Add indirect RAM to Accumulator	1	12
ADD A,#data	Add immediate data to Accumulator	2	12
ADDC A,Rn	Add register to Accumulator with Carry	1	12
ADDC A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC A,#data	Add immediate data to Acc with Carry	2	12
SUBB A,Rn	Subtract Register from Acc with borrow	1	12
SUBB A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB A,#data	Subtract immediate data from Acc with borrow	2	12
INC A	Increment Accumulator	1	12
INC Rn	Increment register	1	12
INC direct	Increment direct byte	2	12
INC @Ri	Increment direct RAM	1	12
DEC A	Decrement Accumulator	1	12
DEC Rn	Decrement Register	1	12
DEC direct	Decrement direct byte	2	12
DEC @Ri	Decrement indirect RAM	1	12

All mnemonics copyrighted ©Intel Corporation 1980

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic	Description	Byte	Oscillator Period	Mnemonic	Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (Continued)				LOGICAL OPERATIONS (Continued)			
INC DPTR	Increment Data Pointer	1	24	RL A	Rotate Accumulator Left	1	12
MUL AB	Multiply A & B	1	48	RLC A	Rotate Accumulator Left through the Carry	1	12
DIV AB	Divide A by B	1	48	RR A	Rotate Accumulator Right	1	12
DA A	Decimal Adjust Accumulator	1	12	RRC A	Rotate Accumulator Right through the Carry	1	12
LOGICAL OPERATIONS				SWAP A	Swap nibbles within the Accumulator	1	12
ANL A,Rn	AND Register to Accumulator	1	12	DATA TRANSFER			
ANL A,direct	AND direct byte to Accumulator	2	12	MOV A,Rn	Move register to Accumulator	1	12
ANL A,@Ri	AND indirect RAM to Accumulator	1	12	MOV A,direct	Move direct byte to Accumulator	2	12
ANL A,#data	AND immediate data to Accumulator	2	12	MOV A,@Ri	Move indirect RAM to Accumulator	1	12
ANL direct,A	AND Accumulator to direct byte	2	12	MOV A,#data	Move immediate data to Accumulator	2	12
ANL direct,#data	AND immediate data to direct byte	3	24	MOV Rn,A	Move Accumulator to register	1	12
ORL A,Rn	OR register to Accumulator	1	12	MOV Rn,direct	Move direct byte to register	2	24
ORL A,direct	OR direct byte to Accumulator	2	12	MOV Rn,#data	Move immediate data to register	2	12
ORL A,@Ri	OR indirect RAM to Accumulator	1	12	MOV direct,A	Move Accumulator to direct byte	2	12
ORL A,#data	OR immediate data to Accumulator	2	12	MOV direct,Rn	Move register to direct byte	2	24
ORL direct,A	OR Accumulator to direct byte	2	12	MOV direct,direct	Move direct byte to direct	3	24
ORL direct,#data	OR immediate data to direct byte	3	24	MOV direct,@Ri	Move indirect RAM to direct byte	2	24
XRL A,Rn	Exclusive-OR register to Accumulator	1	12	MOV direct,#data	Move immediate data to direct byte	3	24
XRL A,direct	Exclusive-OR direct byte to Accumulator	2	12	MOV @Ri,A	Move Accumulator to indirect RAM	1	12
XRL A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12				
XRL A,#data	Exclusive-OR immediate data to Accumulator	2	12				
XRL direct,A	Exclusive-OR Accumulator to direct byte	2	12				
XRL direct,#data	Exclusive-OR immediate data to direct byte	3	24				
CLR A	Clear Accumulator	1	12				
CPL A	Complement Accumulator	1	12				

All mnemonics copyrighted ©Intel Corporation 1980

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic	Description	Byte	Oscillator Period
DATA TRANSFER (Continued)			
MOV @Ri,direct	Move direct byte to indirect RAM	2	24
MOV @Ri,#data	Move immediate data to indirect RAM	2	12
MOV DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX A,DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX @Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX @DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH direct	Push direct byte onto stack	2	24
POP direct	Pop direct byte from stack	2	24
XCH A,Rn	Exchange register with Accumulator	1	12
XCH A,direct	Exchange direct byte with Accumulator	2	12
XCH A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION			
CLR C	Clear Carry	1	12
CLR bit	Clear direct bit	2	12
SETB C	Set Carry	1	12
SETB bit	Set direct bit	2	12
CPL C	Complement Carry	1	12
CPL bit	Complement direct bit	2	12
ANL C,bit	AND direct bit to CARRY	2	24
ANL C,/bit	AND complement of direct bit to Carry	2	24
ORL C,bit	OR direct bit to Carry	2	24
ORL C,/bit	OR complement of direct bit to Carry	2	24
MOV C,bit	Move direct bit to Carry	2	12
MOV bit,C	Move Carry to direct bit	2	24
JC rel	Jump if Carry is set	2	24
JNC rel	Jump if Carry not set	2	24
JB bit,rel	Jump if direct Bit is set	3	24
JNB bit,rel	Jump if direct Bit is Not set	3	24
JBC bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING			
ACALL addr11	Absolute Subroutine Call	2	24
LCALL addr16	Long Subroutine Call	3	24
RET	Return from Subroutine	1	24
RETI	Return from interrupt	1	24
AJMP addr11	Absolute Jump	2	24
LJMP addr16	Long Jump	3	24
SJMP rel	Short Jump (relative addr)	2	24

All mnemonics copyrighted © Intel Corporation 1980

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic	Description	Byte	Oscillator Period	Mnemonic	Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				PROGRAM BRANCHING (Continued)			
JMP @A+DPTR	Jump indirect relative to the DPTR	1	24	CJNE Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
JZ rel	Jump if Accumulator is Zero	2	24	CJNE @Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
JNZ rel	Jump if Accumulator is Not Zero	2	24	DJNZ Rn,rel	Decrement register and Jump if Not Zero	2	24
CJNE A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24	DJNZ direct,rel	Decrement direct byte and Jump if Not Zero	3	24
CJNE A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24	NOP	No Operation	1	12

All mnemonics copyrighted © Intel Corporation 1980

INSTRUCTION DEFINITIONS

ACALL addr11	Function	Description	Bytes	Period	Memory
	Function:	Absolute Call			
	Description:	ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The subroutine called must therefore start within the same 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.			
	Example:	Initially SP equals 07H. The label "SUBRTN" is at program memory location 0345 H. After executing the instruction, ACALL SUBRTN at location 0123H, SP will contain 09H, internal RAM locations 08H and 09H will contain 25H and 01H, respectively, and the PC will contain 0345H.			
	Bytes:	2			
	Cycles:	2			

Encoding:

a10 a9 a8 1 0 0 0 1

a7 a6 a5 a4

a3 a2 a1 a0

Operation:

ACALL
 $(PC) \leftarrow (PC) + 2$
 $(SP) \leftarrow (SP) + 1$
 $((SP)) \leftarrow (PC_{7-0})$
 $(SP) \leftarrow (SP) + 1$
 $((SP)) \leftarrow (PC_{15-8})$
 $(PC_{10-0}) \leftarrow \text{page address}$

ADD A,<src-byte>

Function: Add

Description: ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B). The instruction,

ADD A,R0

will leave 6DH (01101101B) in the Accumulator with the AC flag cleared and both the carry flag and OV set to 1.

ADD A,Rn

Bytes: 1

Cycles: 1

Encoding: 0 0 1 0 1 r r r

Operation: ADD
(A) ← (A) + (Rn)

ADD A,direct

Bytes: 2

Cycles: 1

Encoding: 0 0 1 0 0 1 0 1 direct address

Operation: ADD
(A) ← (A) + (direct)

ADD A,@Ri

Bytes: 1

Cycles: 1

Encoding: 0 0 1 0 0 1 1 1

Operation: ADD
 $(A) \leftarrow (A) + ((R_i))$

ADD A,#data

Bytes: 2

Cycles: 1

Encoding: 0 0 1 0 0 1 0 0 immediate data

Operation: ADD
 $(A) \leftarrow (A) + \#data$

ADDC A,<src-byte>

Function: Add with Carry

Description: ADC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The instruction,

ADDC A,R0

will leave 6EH (01101110B) in the Accumulator with AC cleared and both the Carry flag and OV set to 1.

ADDC A,Rn ALMP addr11

Bytes: 1

Cycles: 1

Encoding: 0 0 1 1 1 r r r

Operation: ADDC
 $(A) \leftarrow (A) + (C) + (R_n)$

ADDC A,direct

Bytes: 2

Cycles: 1

Encoding: 0 0 1 1 0 1 0 1 direct address

Operation: ADDC
 $(A) \leftarrow (A) + (C) + (\text{direct})$

ADDC A,@Ri

Bytes: 1

Cycles: 1

Encoding: 0 0 1 1 0 1 1 i

Operation: ADDC
 $(A) \leftarrow (A) + (C) + ((R_i))$

ADDC A,#data

Bytes: 2

Cycles: 1

Encoding: 0 0 1 1 0 1 0 0 immediate data

Operation: ADDC
 $(A) \leftarrow (A) + (C) + \#data$

AJMP addr11

Function: Absolute Jump

Description: AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (*after* incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2K block of program memory as the first byte of the instruction following AJMP.

Example: The label "JMPADR" is at program memory location 0123H. The instruction,

AJMP JMPADR

is at location 0345H and will load the PC with 0123H.

Bytes: 2

Cycles: 2

Encoding:

a10	a9	a8	0	0	0	1	a7	a6	a5	a4	a3	a2	a1	a0
-----	----	----	---	---	---	---	----	----	----	----	----	----	----	----

Operation: AJMP
 $(PC) \leftarrow (PC) + 2$
 $(PC_{10-0}) \leftarrow \text{page address}$

ANL <dest-byte>, <src-byte>

Function: Logical-AND for byte variables

Description: ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 55H (01010101B) then the instruction,

ANL A,R0

will leave 41H (01000001B) in the Accumulator.

When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The instruction,

ANL P1, #01110011B

will clear bits 7, 3, and 2 of output port 1.

ANL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1 r r r r

Operation:

ANL
(A) ← (A) ∧ (Rn)

ANL A,direct

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1 0 1 0 1

direct address

Operation:

ANL
(A) ← (A) ∧ (direct)

ANL A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1 0 1 1 i

Operation:

ANL
(A) ← (A) ∧ ((Ri))

ANL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1 0 1 0 0

immediate data

Operation:

ANL
(A) ← (A) ∧ #data

ANL direct,A

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1 0 0 1 0

direct address

Operation:

ANL
(direct) ← (direct) ∧ (A)

ANL direct, # data

Bytes: 3

Cycles: 2

Encoding: 0 1 0 1 0 0 1 1 direct address immediate data

Operation: ANL
(direct) ← (direct) ∧ #data

ANL C, <src-bit>

Function: Logical-AND for bit variables

Description: If the Boolean value of the source bit is a logical 0 then clear the carry flag; otherwise leave the carry flag in its current state. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, *but the source bit itself is not affected*. No other flags are affected.

Example: Only direct addressing is allowed for the source operand.
Set the carry flag if, and only if, P1.0 = 1, ACC. 7 = 1, and OV = 0:

MOV C,P1.0 ;LOAD CARRY WITH INPUT PIN STATE

ANL C,ACC.7 ;AND CARRY WITH ACCUM. BIT 7

ANL C,/OV ;AND WITH INVERSE OF OVERFLOW FLAG

ANL C,bit

Bytes: 2

Cycles: 2

Encoding: 1 0 0 0 0 0 1 0 bit address

Operation: ANL
(C) ← (C) ∧ (bit)

ANL C,/bit

Bytes: 2

Cycles: 2

Encoding: 1 0 1 1 0 0 0 0 bit address

Operation: ANL
(C) ← (C) ∧ ¬(bit)

CJNE <dest-byte>, <src-byte>, rel

Function: Compare and Jump if Not Equal.

Description: CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example: The Accumulator contains 34H. Register 7 contains 56H. The first instruction in the sequence,

```

;
; CJNE R7,#60H,NOT_EQ
; R7 = 60H.
NOT_EQ: JC REQ_LOW ; IF R7 < 60H.
; R7 > 60H.
;

```

sets the carry flag and branches to the instruction at label NOT_EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to Port 1 is also 34H, then the instruction,

WAIT: CJNE A,P1,WAIT

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34H.)

CJNE A,direct,rel

Bytes: 3

Cycles: 2

Encoding:

1	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

direct address

rel. address

Operation:

```
(PC) ← (PC) + 3
IF (A) <> (direct)
THEN
```

$$(\text{PC}) \leftarrow (\text{PC}) + \text{relative offset}$$

IF (A) < (*direct*)

THEN

(C) ← 1

ELSE

$$(C) \leftarrow 0$$

CJNE A, #data, rel

Bytes: 3

Cycles: 2

Encoding:

1	0	1	1
---	---	---	---

0	1	0	0
---	---	---	---

immediate data			
----------------	--	--	--

rel. address			
--------------	--	--	--

Operation: $(PC) \leftarrow (PC) + 3$
 IF (A) <> data
 THEN
 $(PC) \leftarrow (PC) + \text{relative offset}$
 ELSE
 $(C) \leftarrow 1$
 ELSE
 $(C) \leftarrow 0$

CJNE Rn, #data, rel

Bytes: 3

Cycles: 2

Encoding:

1	0	1	1
---	---	---	---

1	r	r	r
---	---	---	---

immediate data			
----------------	--	--	--

rel. address			
--------------	--	--	--

Operation: $(PC) \leftarrow (PC) + 3$
 IF (Rn) <> data
 THEN
 $(PC) \leftarrow (PC) + \text{relative offset}$
 ELSE
 $(C) \leftarrow 1$
 ELSE
 $(C) \leftarrow 0$

CJNE @Ri, #data, rel

Bytes: 3

Cycles: 2

Encoding:

1	0	1	1
---	---	---	---

0	1	1	i
---	---	---	---

immediate data			
----------------	--	--	--

rel. address			
--------------	--	--	--

Operation: $(PC) \leftarrow (PC) + 3$
 IF ((Ri)) <> data
 THEN
 $(PC) \leftarrow (PC) + \text{relative offset}$
 ELSE
 $(C) \leftarrow 1$
 ELSE
 $(C) \leftarrow 0$

CLR A

CLR A

Function: Clear Accumulator

Complement Accumulator

Function:

Description: The Accumulator is cleared (all bits set on zero). No flags are affected.

Description:

Example: The Accumulator contains 5CH (01011100B). The instruction,

Example:

CLR A

CPL A

will leave the Accumulator set to 00H (00000000B).

CPL A

Bytes: 1

Bytes:

Cycles: 1

Cycles:

Encoding: 1 1 1 1 0 1 0 0

1 1 1 1 0 1 0 0

Encoding:

Operation: CLR
(A) ← 0

CPL
(A) ← ¬(A)

Operation:

CLR bit

CLR bit

Function: Clear bit

Complement bit

Function:

Description: The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Description:

Example: Port 1 has previously been written with 5DH (01011101B). The instruction,

CLR P1.2

CPL P1.1

CPL P1.2

will leave the port set to 59H (01011001B).

Example:

CLR C

Bytes: 1

Cycles: 1

Encoding: 1 1 0 0 0 0 1 1

Operation: CLR
(C) ← 0

CPL C

Bytes:

Cycles:

CLR bit

Bytes: 2

Cycles: 1

Encoding: 1 1 0 0 0 0 1 0

bit address

Operation: CLR
(bit) ← 0

1 1 0 1 0 0 1 1

Encoding:

Operation:

CPL A

Function: Complement Accumulator

Description: Each bit of the Accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to a zero and vice-versa. No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The instruction,

CPL A

will leave the Accumulator set to 0A3H (10100011B).

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	0 1 0 0
---------	---------

Operation: CPL
(A) ← ¬(A)

CPL bit

Function: Complement bit

Description: The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.

Note: When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: Port 1 has previously been written with 5BH (01011101B). The instruction sequence,

CPL P1.1

CPL P1.2

will leave the port set to 5BH (01011101B).

CPL C

Bytes: 1

Cycles: 1

Encoding:

1 0 1 1	0 0 1 1
---------	---------

Operation: CPL
(C) ← ¬(C)

CPL bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 1 0 0 1 0

bit address

Operation:

CPL

(bit) ← (bit)

DA A

Function: Decimal-adjust Accumulator for Addition

Description: DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

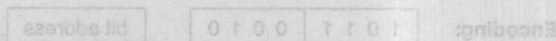
If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx-111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carry-out of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the Accumulator, depending on initial Accumulator and PSW conditions.

Note: DA A cannot simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DA A apply to decimal subtraction.

Example: The Accumulator holds the value 56H (01010110B) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67H (01100111B) representing the packed BCD digits of the decimal number 67. The carry flag is set. The instruction sequence.

ADDC A,R3
DA A



will first perform a standard two's-complement binary addition, resulting in the value 0BEH (10111110) in the Accumulator. The carry and auxiliary carry flags will be cleared.

The Decimal Adjust instruction will then alter the Accumulator to the value 24H (00100100B), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag will be set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01H or 99H. If the Accumulator initially holds 30H (representing the digits of 30 decimal), then the instruction sequence,

ADD A,#99H

DA A

will leave the carry set and 29H in the Accumulator, since $30 + 99 = 129$. The low-order byte of the sum can be interpreted to mean $30 - 1 = 29$.

Bytes: 1

Cycles: 1

Encoding: 1101 0100

Operation: DA A
-contents of Accumulator are BCD
IF $[(A_{3.0}) > 9] \vee [(AC) = 1]$
THEN $(A_{3.0}) \leftarrow (A_{3.0}) + 6$
AND

IF $[(A_{7.4}) > 9] \vee [(C) = 1]$
THEN $(A_{7.4}) \leftarrow (A_{7.4}) + 6$

DEC byte

Function: Decrement

Description: The variable indicated is decremented by 1. An original value of 00H will underflow to 0FFH. No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Register 0 contains 7FH (01111111B). Internal RAM locations 7EH and 7FH contain 00H and 40H, respectively. The instruction sequence,

DEC @R0

DEC R0

DEC @R0

will leave register 0 set to 7EH and internal RAM locations 7EH and 7FH set to 0FFH and 3FH.

DEC A

Bytes: 1

Cycles: 1

Encoding: 0001 0100

Operation: DEC
(A) ← (A) - 1

DEC Rn

Bytes: 1

Cycles: 1

Encoding: 0001 1rrr

Operation: DEC
(Rn) ← (Rn) - 1

DEC direct

Bytes: 2

Cycles: 1

Encoding: 0 0 0 1 0 1 0 1 direct address

Operation: DEC
 $(\text{direct}) \leftarrow (\text{direct}) - 1$

DEC @Ri

Bytes: 1

Cycles: 1

Encoding: 0 0 0 1 0 1 1 i

Operation: DEC
 $((Ri)) \leftarrow ((Ri)) - 1$

DIV AB

Function: Divide

Description: DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags will be cleared.

Exception: if B had originally contained 00H, the values returned in the Accumulator and B-register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

Example: The Accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The instruction,

DIV AB

will leave 13 in the Accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since $251 = (13 \times 18) + 17$. Carry and OV will both be cleared.

Bytes: 1

Cycles: 4

Encoding: 1 0 0 0 0 1 0 0

Operation: DIV
 $(A)_{15-8} \leftarrow (A)/(B)$
 $(B)_{7-0}$

DJNZ <byte>, <rel-addr>

Function: Decrement and Jump if Not Zero

Description: DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00H will underflow to 0FFH. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Internal RAM locations 40H, 50H, and 60H contain the values 01H, 70H, and 15H, respectively. The instruction sequence,

```
DJNZ 40H, LABEL_1
DJNZ 50H, LABEL_2
DJNZ 60H, LABEL_3
```

will cause a jump to the instruction at label LABEL_2 with the values 00H, 6FH, and 15H in the three RAM locations. The first jump was *not* taken because the result was zero.

This instruction provides a simple way of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
TOGGLE: MOV R2, #8
        CPL P1.7
        DJNZ R2, TOGGLE
```

will toggle P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1. Each pulse will last three machine cycles; two for DJNZ and one to alter the pin.

DJNZ Rn,rel

Bytes: 2

Cycles: 2

Encoding:

1 1 0 1 1 r r r

rel. address

0 0 1 0 0 0 0 0

Operation:

```
DJNZ
(PC) ← (PC) + 2
(Rn) ← (Rn) - 1
IF (Rn) > 0 or (Rn) < 0
THEN
    (PC) ← (PC) + rel
```

DJNZ direct,rel

Bytes: 3

Cycles: 2

Encoding:

1 1 0 1

0 1 0 1

direct address

rel. address

Operation:

DJNZ

(PC) ← (PC) + 2

(direct) ← (direct) - 1

IF (direct) > 0 or (direct) < 0

THEN

(PC) ← (PC) + rel

INC <byte>

Function: Increment

Description:

INC increments the indicated variable by 1. An original value of 0FFH will overflow to 00H. No flags are affected. Three addressing modes are allowed: register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example:

Register 0 contains 7EH (01111110B). Internal RAM locations 7EH and 7FH contain 0FFH and 40H, respectively. The instruction sequence,

INC @R0

INC R0

INC @R0

will leave register 0 set to 7FH and internal RAM locations 7EH and 7FH holding (respectively) 00H and 41H.

INC A

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0

0 1 0 0

rel. address

1 1 1 1

Operation:

INC

(A) ← (A) + 1

INC Rn		1B bit,rel
Bytes:	1	
Cycles:	1	
Encoding:	0 0 0 0 1 r r r	
Operation:	INC (Rn) ← (Rn) + 1	
INC direct		
Bytes:	2	
Cycles:	1	
Encoding:	0 0 0 0 0 1 0 1 direct address	
Operation:	INC (direct) ← (direct) + 1	
INC @Ri		
Bytes:	1	
Cycles:	1	
Encoding:	0 0 0 0 0 1 1 i	
Operation:	INC ((Ri)) ← ((Ri)) + 1	
INC DPTR		
Function:	Increment Data Pointer	
Description:	Increment the 16-bit data pointer by 1. A 16-bit increment (modulo 2 ¹⁶) is performed; an overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will increment the high-order byte (DPH). No flags are affected.	
	This is the only 16-bit register which can be incremented.	
Example:	Registers DPH and DPL contain 12H and 0FEH, respectively. The instruction sequence,	
	<pre> INC DPTR INC DPTR INC DPTR </pre>	
	will change DPH and DPL to 13H and 01H.	
Bytes:	1	
Cycles:	2	
Encoding:	1 0 1 0 0 0 1 1	
Operation:	INC (DPTR) ← (DPTR) + 1	

JB bit,rel

Function: Jump if Bit set

Description: If the indicated bit is a one, jump to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56 (01010110B). The instruction sequence,

JB P1.2,LABEL1

JB ACC.2,LABEL2

will cause program execution to branch to the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0 0 1 0 0 0 0 0

bit address

rel. address

Operation:

JB

(PC) ← (PC) + 3

IF (bit) = 1

THEN

(PC) ← (PC) + rel

JBC bit,rel

Function: Jump if Bit is set and Clear bit

Description: If the indicated bit is one, branch to the address indicated; otherwise proceed with the next instruction. *The bit will not be cleared if it is already a zero.* The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.

Note: When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: The Accumulator holds 56H (01010110B). The instruction sequence,

JBC ACC.3,LABEL1

JBC ACC.2,LABEL2

will cause program execution to continue at the instruction identified by the label LABEL2, with the Accumulator modified to 52H (01010010B).

Bytes:	3				
Cycles:	2				
Encoding:	0 0 0 1	0 0 0 0	bit address	rel. address	
Operation:	JBC $(PC) \leftarrow (PC) + 3$ IF (bit) = 1 THEN $(bit) \leftarrow 0$ $(PC) \leftarrow (PC) + rel$				
JC rel					

Function: Jump if Carry is set

Description: If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example: The carry flag is cleared. The instruction sequence,

```
JC LABEL1
CPL C
JC LABEL2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding: 0 1 0 0 0 0 0 0 rel. address

Operation: JC
 $(PC) \leftarrow (PC) + 2$
 IF (C) = 1
 THEN
 $(PC) \leftarrow (PC) + rel$

JMP @A + DPTR

Function: Jump indirect

Description: Add the eight-bit unsigned contents of the Accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

Example: An even number from 0 to 6 is in the Accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```

MOV    DPTR, #JMP_TBL
JMP    @A + DPTR
JMP_TBL: AJMP LABEL0
        AJMP LABEL1
        AJMP LABEL2
        AJMP LABEL3
    
```

If the Accumulator equals 04H when starting this sequence, execution will jump to label LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

Bytes: 1

Cycles: 2

Encoding:

0	1	1	1
---	---	---	---

0	0	1	1
---	---	---	---

Operation: JMP
 $(PC) \leftarrow (A) + (DPTR)$

JNB bit,rel

Function: Jump if Bit Not set

Description: If the indicated bit is a zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56H (01010110B). The instruction sequence,

```
JNB P1.3,LABEL1
JNB ACC.3,LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding: 0 0 1 1 0 0 0 0 bit address rel. address 0

Operation: JNB
 $(PC) \leftarrow (PC) + 3$
 IF (bit) = 0
 THEN $(PC) \leftarrow (PC) + \text{rel.}$

JNC rel

Function: Jump if Carry not set

Description: If the carry flag is a zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example: The carry flag is set. The instruction sequence,

```
JNC LABEL1
CPL C
JNC LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding: 0 1 0 1 0 0 0 0 rel. address

Operation: JNC
 $(PC) \leftarrow (PC) + 2$
 IF (C) = 0
 THEN $(PC) \leftarrow (PC) + \text{rel}$

JNZ rel

Function: Jump if Accumulator Not Zero

Description: If any bit of the Accumulator is a one, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 00H. The instruction sequence,

```
JNZ LABEL1
INC A
JNZ LABEL2
```

will set the Accumulator to 01H and continue at label LABEL2.

Bytes: 2

Cycles: 2

Encoding: 0 1 1 1 0 0 0 0 rel. address

Operation: JNZ
 $(PC) \leftarrow (PC) + 2$
 IF $(A) \neq 0$
 THEN $(PC) \leftarrow (PC) + \text{rel}$

JZ rel

Function: Jump if Accumulator Zero

Description: If all bits of the Accumulator are zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally contains 01H. The instruction sequence,

```
JZ LABEL1
DEC A
JZ LABEL2
```

will change the Accumulator to 00H and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding: 0 1 1 0 0 0 0 0 rel. address

Operation: JZ
 $(PC) \leftarrow (PC) + 2$
 IF $(A) = 0$
 THEN $(PC) \leftarrow (PC) + \text{rel}$

LCALL addr16

Function: Long call

Description: LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64K-byte program memory address space. No flags are affected.

Example: Initially the Stack Pointer equals 07H. The label "SUBRTN" is assigned to program memory location 1234H. After executing the instruction,

LCALL SUBRTN

at location 0123H, the Stack Pointer will contain 09H, internal RAM locations 08H and 09H will contain 26H and 01H, and the PC will contain 1234H.

Bytes: 3

Cycles: 2

Encoding:

0 0 0 1 0 0 1 0

addr15-addr8

addr7-addr0

Operation:

LCALL
 $(PC) \leftarrow (PC) + 3$
 $(SP) \leftarrow (SP) + 1$
 $((SP)) \leftarrow (PC_{7-0})$
 $(SP) \leftarrow (SP) + 1$
 $((SP)) \leftarrow (PC_{15-8})$
 $(PC) \leftarrow \text{addr}_{15-0}$

LJMP addr16

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64K program memory address space. No flags are affected.

Example: The label "JMPADR" is assigned to the instruction at program memory location 1234H. The instruction,

LJMP JMPADR

at location 0123H will load the program counter with 1234H.

Bytes: 3

Cycles: 2

Encoding:

0 0 0 0 0 0 1 0

addr15-addr8

addr7-addr0

Operation:

LJMP
 $(PC) \leftarrow \text{addr}_{15-0}$

MOV <dest-byte>, <src-byte>

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

Example: Internal RAM location 30H holds 40H. The value of RAM location 40H is 10H. The data present at input port 1 is 11001010B (0CAH).

```
MOV R0, #30H ;R0 <= 30H
MOV A, @R0 ;A <= 40H
MOV R1, A ;R1 <= 40H
MOV B, @R1 ;B <= 10H
MOV @R1, P1 ;RAM (40H) <= 0CAH
MOV P2, P1 ;P2 #0CAH
```

leaves the value 30H in register 0, 40H in both the Accumulator and register 1, 10H in register B, and 0CAH (11001010B) both in RAM location 40H and output on port 2.

MOV A, Rn

Bytes: 1

Cycles: 1

Encoding: 1 1 1 0 1 r r r

Operation: MOV
(A) ← (Rn)

***MOV A, direct**

Bytes: 2

Cycles: 1

Encoding: 1 1 1 0 0 1 0 1 direct address

Operation: MOV
(A) ← (direct)

MOV A, ACC is not a valid instruction.

1 1 1 1	0 1 0 1
---------	---------

MOV
(direct) ← (A)

1 0 0 0	1 r r r
---------	---------

MOV
(direct) ← (Rn)

1 0 0 0	0 1 0 1
---------	---------

dir. addr. (dest

MOV
(direct) ← (direct)

1 0 0 0	0 1 1 i
---------	---------

MOV
(direct) ← ((Ri))

0 1 1 1	0 1 0 1
---------	---------

immediate data

MOV
(direct) ← #data

MOV @Ri,A

Bytes: 1
Cycles: 1

Encoding: 1 1 1 1 0 1 1 i

Operation: MOV ((Ri)) ← (A)

MOV @Ri,direct

Bytes: 2
Cycles: 2

Encoding: 1 0 1 0 0 1 1 i

Operation: MOV ((Ri)) ← (direct)

MOV @Ri,#data

Bytes: 2
Cycles: 1

Encoding: 0 1 1 1 0 1 1 i

Operation: MOV ((Ri)) ← #data

MOV <dest-bit>,<src-bit>

Function: Move bit data

Description: The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: The carry flag is originally set. The data present at input Port 3 is 11000101B. The data previously written to output Port 1 is 35H (00110101B).

MOV P1.3,C
MOV C,P3.3
MOV P1.2,C

will leave the carry cleared and change Port 1 to 39H (00111001B).

MOV C,bit

Bytes: 2

Cycles: 1

Encoding: 1 0 1 0 0 0 1 0 bit address

Operation: MOV (C) ← (bit)

MOV bit,C

Bytes: 2

Cycles: 2

Encoding: 1 0 0 1 0 0 1 0 bit address

Operation: MOV (bit) ← (C)

MOV DPTR,#data16

Function: Load Data Pointer with a 16-bit constant

Description: The Data Pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.

This is the only instruction which moves 16 bits of data at once.

Example: The instruction,

MOV DPTR,#1234H

will load the value 1234H into the Data Pointer: DPH will hold 12H and DPL will hold 34H.

Bytes: 3

Cycles: 2

Encoding: 1 0 0 1 0 0 0 0 immed. data15-8 immed. data7-0

Operation: MOV (DPTR) ← #data₁₅₋₀
DPH ← #data₁₅₋₈ DPL ← #data₇₋₀

MOVC A,@A + <base-reg>
MOVC <dest-byte>, <src-byte>
Function: Move Code byte

Function: Move External

Description: The MOVC instructions load the Accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit Accumulator contents and the contents of a sixteen-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

Example: A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive.

```
REL_PC: INC A
        MOVC A,@A+PC
        RET

        DB 66H
        DB 77H
        DB 88H
        DB 99H
```

If the subroutine is called with the Accumulator equal to 01H, it will return with 77H in the Accumulator. The INC A before the MOVC instruction is needed to "get around" the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the Accumulator instead.

MOVC A,@A + DPTR
Bytes: 1

Cycles: 2

Encoding:

1	0	0	1
0	0	1	1

Operation: MOVC
 $(A) \leftarrow ((A) + (DPTR))$
MOVC A,@A + PC
Bytes: 1

Cycles: 2

Encoding:

1	0	0	0
0	0	1	1

Operation: MOVC
 $(PC) \leftarrow (PC) + 1$
 $(A) \leftarrow ((A) + (PC))$

MOVX <dest-byte>, <src-byte>

MOVX A, @A+ <disp-reg>

Function: Move External

Function: Move Code byte

Description: The MOVX instructions transfer data between the Accumulator and a byte of external data memory, hence the "X" appended to MOV. There are two types of instructions, differing in whether they provide an eight-bit or sixteen-bit indirect address to the external data RAM.

In the first type, the contents of R0 or R1 in the current register bank provide an eight-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, any output port pins can be used to output higher-order address bits. These pins would be controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, the Data Pointer generates a sixteen-bit address. P2 outputs the high-order eight address bits (the contents of DPH) while P0 multiplexes the low-order eight bits (DPL) with data. The P2 Special Function Register retains its previous contents while the P2 output buffers are emitting the contents of DPH. This form is faster and more efficient when accessing very large data arrays (up to 64K bytes), since no additional instructions are needed to set up the output ports.

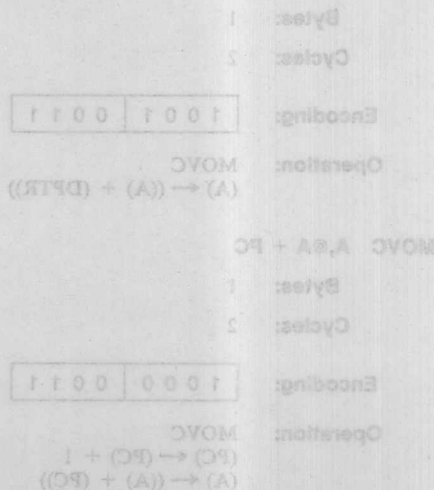
It is possible in some situations to mix the two MOVX types. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2 followed by a MOVX instruction using R0 or R1.

Example: An external 256 byte RAM using multiplexed address/data lines (e.g., an Intel 8155 RAM/I/O/Timer) is connected to the 8051 Port 0. Port 3 provides control lines for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12H and 34H. Location 34H of the external RAM holds the value 56H. The instruction sequence,

MOVX A, @R1

MOVX @R0, A

copies the value 56H into both the Accumulator and external RAM location 12H.



MOVX A,@Ri		NOP	
Bytes:	1	Function: No Operation	
Cycles:	2	Description: Execution continues at the following instruction. Other than the PC, no registers or flags are affected.	
Encoding:	1 1 1 0 0 0 1 i	Example: It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 2 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse. This may be done (assuming no interrupts) with the instruction sequence:	
Operation:	MOVX (A) ← ((Ri))		
MOVX A,@DPTR		CLR P2.7 NOP NOP NOP NOP SETB P2.7	
Bytes:	1	Bytes: 1	
Cycles:	2	Cycles: 1	
Encoding:	1 1 1 0 0 0 0 0	Encoding: 0 0 0 0 0 0 0 0	
Operation:	MOVX (A) ← ((DPTR))	Operation: NOP (PC) ← (PC) + 1	
MOVX @Ri,A		MUL AB	
Bytes:	1	Function: Multiply	
Cycles:	2	Description: MUL AB multiplies the unsigned eight-bit integers in the Accumulator and the high-order byte of the sixteen-bit product is left in the Accumulator, and the low-order byte is left in the B register. If the product is greater than 255 (0FFH) the overflow flag is set. 1. The carry flag is always cleared.	
Encoding:	1 1 1 1 0 0 1 i	Example: Originally the Accumulator holds the value 80 (20H). Register B holds the value 150 (96H). The instruction:	
Operation:	MOVX ((Ri)) ← (A)	MUL AB	
MOVX @DPTR,A		will give the product 12,800 (3200H), so B is changed to 255 (0FFH) and the Accumulator is cleared. The overflow flag is set, carry is cleared.	
Bytes:	1	Bytes: 1	
Cycles:	2	Cycles: 4	
Encoding:	1 1 1 1 0 0 0 0	Encoding: 1 0 1 0 0 1 0 0	
Operation:	MOVX (DPTR) ← (A)	Operation: MUL (A) ← ((A) X (B)) (B)15-8	

NOP

Function: No Operation

Description: Execution continues at the following instruction. Other than the PC, no registers or flags are affected.

Example: It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming no interrupts are enabled) with the instruction sequence,

CLR P2.7

NOP

NOP

NOP

NOP

SETB P2.7

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0
---	---	---	---

0	0	0	0
---	---	---	---

Operation: NOP
(PC) \leftarrow (PC) + 1

MUL AB

Function: Multiply

Description: MUL AB multiplies the unsigned eight-bit integers in the Accumulator and register B. The low-order byte of the sixteen-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (0FFH) the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.

Example: Originally the Accumulator holds the value 80 (50H). Register B holds the value 160 (0A0H). The instruction,

MUL AB

will give the product 12,800 (3200H), so B is changed to 32H (00110010B) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes: 1

Cycles: 4

Encoding:

1	0	1	0
---	---	---	---

0	1	0	0
---	---	---	---

Operation: MUL
(A)₇₋₀ \leftarrow (A) X (B)
(B)₁₅₋₈

Function: Logical-OR for byte variables

Description: ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and R0 holds 55H (01010101B) then the instruction,

ORL A,R0

will leave the Accumulator holding the value 0D7H (11010111B).

When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction,

ORL P1,#00110010B

will set bits 5, 4, and 1 of output Port 1.

ORL A,Rn

Bytes: 1

Cycles: 1

Encoding: 0 1 0 0 1 r r r

Operation: ORL
(A) ← (A) V (Rn)

ORL A,direct	< 8-bit byte > < 8-bit byte >	
Bytes: 2	Function: Logical-OR for byte variables	
Cycles: 1	Description: ORL performs the bitwise logical-OR operation between the indicated variables. The results in the destination are stored back in the destination.	
Encoding:	0 1 0 0 0 1 0 1	direct address
Operation:	ORL (A) ← (A) V (direct)	
ORL A,@Ri	Example: If the Accumulator holds 0C3H (11000111B) and R0 holds 25H (00101011B), the output data will be read from the output data latch, not the input port.	
Bytes: 1	Function: Logical-OR for byte variables	
Cycles: 1	Description: ORL performs the bitwise logical-OR operation between the indicated variables. The results in the destination are stored back in the destination.	
Encoding:	0 1 0 0 0 1 1 i	
Operation:	ORL (A) ← (A) V ((Ri))	
ORL A,#data	When the destination is a directly addressed byte in any RAM location or hardware register, the pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable in the Accumulator at run-time. The instruction will set bits 5, 4, and 0 of the Accumulator holding the value 0D7H (11010111B).	
Bytes: 2	Function: Logical-OR for byte variables	
Cycles: 1	Description: ORL performs the bitwise logical-OR operation between the indicated variables. The results in the destination are stored back in the destination.	
Encoding:	0 1 0 0 0 1 0 0	immediate data
Operation:	ORL (A) ← (A) V #data	
ORL direct,A	Function: Logical-OR for byte variables	
Bytes: 2	Description: ORL performs the bitwise logical-OR operation between the indicated variables. The results in the destination are stored back in the destination.	
Cycles: 1	Encoding: 0 1 0 0 0 1 0 0	
Encoding:	0 1 0 0 0 0 1 0	direct address
Operation:	ORL (direct) ← (direct) V (A)	
ORL direct,#data	Function: Logical-OR for byte variables	
Bytes: 3	Description: ORL performs the bitwise logical-OR operation between the indicated variables. The results in the destination are stored back in the destination.	
Cycles: 2	Encoding: 0 1 0 0 0 0 1 1	
Encoding:	0 1 0 0 0 0 1 1	direct addr. immediate data
Operation:	ORL (direct) ← (direct) V #data	

ORL C, <src-bit>

POP direct

Function: Logical-OR for bit variables

Function: Pop from stack

Description: Set the carry flag if the Boolean value is a logical 1; leave the carry in its current state otherwise. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Example: Set the carry flag if and only if P1.0 = 1, ACC. 7 = 1, or OV = 0:

MOV C,P1.0 ;LOAD CARRY WITH INPUT PIN P10

ORL C,ACC.7 ;OR CARRY WITH THE ACC. BIT 7

ORL C,/OV ;OR CARRY WITH THE INVERSE OF OV.

ORL C,bit

POP SP

Bytes: 2

Cycles: 2

Encoding: 0 1 1 1 0 0 1 0 bit address

Operation: ORL
(C) ← (C) V (bit)

direct address

1 1 0 1 0 0 0 0

ORL C,/bit

POP

Bytes: 2

Cycles: 2

Encoding: 1 0 1 0 0 0 0 0 bit address

Operation: ORL
(C) ← (C) V (bit)

(SP) ← (SP) - 1
(direct) ← (SP)

Operation:

PUSH direct

Push onto stack

Description: The Stack Pointer is incremented by one. The contents of the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine the Stack Pointer contains 00H. The Data Pointer holds the value 0123H. The instruction sequence:

PUSH DPL

PUSH DPH

will leave the Stack Pointer set to 00H and store 23H and 01H in internal RAM locations 00H and 01H, respectively.

Bytes: 2

Cycles: 2

direct address

1 1 0 0 0 0 0 0

PUSH
(SP) ← (SP) + 1
(SP) ← (direct)

Operation:

POP direct

Function: Pop from stack.

Description: The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

Example: The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The instruction sequence,

```
POP DPH
```

```
POP DPL
```

will leave the Stack Pointer equal to the value 30H and the Data Pointer set to 0123H. At this point the instruction,

```
POP SP
```

will leave the Stack Pointer set to 20H. Note that in this special case the Stack Pointer was decremented to 2FH before being loaded with the value popped (20H).

Bytes: 2

Cycles: 2

Encoding: 1 1 0 1 0 0 0 0

direct address

Operation: POP
(direct) ← ((SP))
(SP) ← (SP) - 1

PUSH direct

Function: Push onto stack

Description: The Stack Pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The instruction sequence,

```
PUSH DPL
```

```
PUSH DPH
```

will leave the Stack Pointer set to 0BH and store 23H and 01H in internal RAM locations 0AH and 0BH, respectively.

Bytes: 2

Cycles: 2

Encoding: 1 1 0 0 0 0 0 0

direct address

Operation: PUSH
(SP) ← (SP) + 1
((SP)) ← (direct)

RET

Function: Return from subroutine

Description: RET pops the high- and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by two. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.

Example: The Stack Pointer originally contains the value 0BH. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

RET

will leave the Stack Pointer equal to the value 09H. Program execution will continue at location 0123H.

Bytes: 1

Cycles: 2

Encoding: 0 0 1 0 0 0 1 0

Operation: RET
 $(PC_{15-8}) \leftarrow ((SP))$
 $(SP) \leftarrow (SP) - 1$
 $(PC_{7-0}) \leftarrow ((SP))$
 $(SP) \leftarrow (SP) - 1$

RETI

Function: Return from interrupt

Description: RETI pops the high- and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected; the PSW is *not* automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower- or same-level interrupt had been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.

Example: The Stack Pointer originally contains the value 0BH. An interrupt was detected during the instruction ending at location 0122H. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

RETI

will leave the Stack Pointer equal to 09H and return program execution to location 0123H.

Bytes: 1

Cycles: 2

Encoding: 0 0 1 1 0 0 1 0

Operation: RETI
 $(PC_{15-8}) \leftarrow ((SP))$
 $(SP) \leftarrow (SP) - 1$
 $(PC_{7-0}) \leftarrow ((SP))$
 $(SP) \leftarrow (SP) - 1$

Function: Rotate Accumulator Left

Description: The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,
RL A

leaves the Accumulator holding the value 8BH (10001011B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Operation:

RL

$(A_n + 1) \leftarrow (A_n) \quad n = 0 - 6$
 $(A0) \leftarrow (A7)$

RLC A

Function: Rotate Accumulator Left through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,
RLC A

leaves the Accumulator holding the value 8BH (10001010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Operation:

RLC

$(A_n + 1) \leftarrow (A_n) \quad n = 0 - 6$
 $(A0) \leftarrow (C)$
 $(C) \leftarrow (A7)$

RR A

Function: Rotate Accumulator Right

Description: The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

RR A

leaves the Accumulator holding the value 0E2H (11100010B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0
---	---	---	---

0	0	1	1
---	---	---	---

Operation: RR
 $(A_n) \leftarrow (A_n + 1) \quad n = 0 - 6$
 $(A_7) \leftarrow (A_0)$

RRC A

Function: Rotate Accumulator Right through Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original value of the carry flag moves into the bit 7 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), the carry is zero. The instruction,

RRC A

leaves the Accumulator holding the value 62 (01100010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1
---	---	---	---

0	0	1	1
---	---	---	---

Operation: RRC
 $(A_n) \leftarrow (A_n + 1) \quad n = 0 - 6$
 $(A_7) \leftarrow (C)$
 $(C) \leftarrow (A_0)$

SETB <bit>

Function: Set Bit

Description: SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

Example: The carry flag is cleared. Output Port 1 has been written with the value 34H (00110100B). The instructions,

SETB C

SETB P1.0

will leave the carry flag set to 1 and change the data output on Port 1 to 35H (00110101B).

SETB C

Bytes: 1

Cycles: 1

Encoding: 1 1 0 1 0 0 1 1

Operation: SETB
(C) ← 1

SETB bit

Bytes: 2

Cycles: 1

Encoding: 1 1 0 1 0 0 1 0 bit address

Operation: SETB
(bit) ← 1

SJMP rel

<subb A, >src-byte>

Function: Short Jump

Function: Subtract with borrow

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Example: The label "RELADR" is assigned to an instruction at program memory location 0123H. The instruction,

SJMP RELADR

will assemble into location 0100H. After the instruction is executed, the PC will contain the value 0123H.

(Note: Under the above conditions the instruction following SJMP will be at 102H. Therefore, the displacement byte of the instruction will be the relative offset (0123H-0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

Bytes: 2

Cycles: 2

Encoding:

1	0	0	0	0	0	0	0	rel. address
---	---	---	---	---	---	---	---	--------------

Operation:

SJMP

$(PC) \leftarrow (PC) + 2$

$(PC) \leftarrow (PC) + \text{rel}$

Function: Subtract with borrow

Description: SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set *before* executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. The instruction,

SUBB A,R2

will leave the value 74H (01110100B) in the accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9H minus 54H is 75H. The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR C instruction.

SUBB A,Rn

Bytes: 1

Cycles: 1

Encoding:

1	0	0	1
---	---	---	---

1	r	r	r
---	---	---	---

Operation: SUBB
 $(A) \leftarrow (A) - (C) - (Rn)$

SUBB A, direct

Bytes: 2

Cycles: 1

Encoding:

1 0 0 1 0 1 0 1

direct address

Operation:

SUBB

 $(A) \leftarrow (A) - (C) - (\text{direct})$

SUBB A, @Ri

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1 0 1 1 i

Operation:

SUBB

 $(A) \leftarrow (A) - (C) - ((Ri))$

SUBB A, #data

Bytes: 2

Cycles: 1

Encoding:

1 0 0 1 0 1 0 0

immediate data

Operation:

SUBB

 $(A) \leftarrow (A) - (C) - \#data$

SWAP A

Function: Swap nibbles within the Accumulator

Description: SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a four-bit rotate instruction. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

SWAP A

leaves the Accumulator holding the value 5CH (01011100B).

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0 0 1 0 0

Operation:

SWAP

 $(A_{3-0}) \leftrightarrow (A_{7-4})$

XCH A,<byte>

Function: Exchange Accumulator with byte variable

Description: XCH loads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

Example: R0 contains the address 20H. The Accumulator holds the value 3FH (00111111B). Internal RAM location 20H holds the value 75H (01101010B). The instruction,

XCH A,@R0

will leave RAM location 20H holding the values 3FH (00111111B) and 75H (01101010B) in the accumulator.

XCH A,Rn

Bytes: 1

Cycles: 1

Encoding: 1 1 0 0 1 r r r

Operation: XCH
(A) ↔ (Rn)

XCH A,direct

Bytes: 2

Cycles: 1

Encoding: 1 1 0 0 0 1 0 1 direct address

Operation: XCH
(A) ↔ (direct)

XCH A,@Ri

Bytes: 1

Cycles: 1

Encoding: 1 1 0 0 0 1 1 i

Operation: XCH
(A) ↔ ((Ri))

XCHD A,@Ri

Function: Exchange Digit

Description: XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected.

Example: R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (0110101B). The instruction,

XCHD A,@R0

will leave RAM location 20H holding the value 76H (0110110B) and 35H (00110101B) in the Accumulator.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1
0	1	1	1

Operation: XCHD
(A₃₋₀) ↔ ((Ri₃₋₀))

XRL <dest-byte>,<src-byte>

Function: Logical Exclusive-OR for byte variables

Description: XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

(Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.)

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

XRL A,R0

will leave the Accumulator holding the value 69H (01101001B).

When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The instruction,

XRL P1,#00110001B

will complement bits 5, 4, and 0 of output Port 1.

XRL A,Rn		XCHD A,Rn	
Bytes:	1	Function:	Exchange Digit
Cycles:	1	Description:	XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal register specified by the instruction. The high-order nibble (bits 7-4) is not affected.
Encoding:	0 1 1 0 1 r r r	Example:	R0 contains the address 20H. The Accumulator holds the value 75H (01101010B). The instruction XCHD A,R0 will leave RAM location 20H holding the value 75H (01101010B) and 75H (01101010B) in the Accumulator.
Operation:	XRL (A) ← (A) ∨ (Rn)		
XRL A,direct		XCHD A,R0	
Bytes:	2	Function:	Exchange Digit
Cycles:	1	Description:	XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal register specified by the instruction. The high-order nibble (bits 7-4) is not affected.
Encoding:	0 1 1 0 0 1 0 1	Example:	R0 contains the address 20H. The Accumulator holds the value 75H (01101010B). The instruction XCHD A,R0 will leave RAM location 20H holding the value 75H (01101010B) and 75H (01101010B) in the Accumulator.
		Bytes:	1
		Cycles:	1
Operation:	XRL (A) ← (A) ∨ (direct)	Encoding:	1 1 0 1 0 1 1 i
XRL A,@Ri		XCHD A,R0	
Bytes:	1	Function:	Exchange Digit
Cycles:	1	Description:	XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal register specified by the instruction. The high-order nibble (bits 7-4) is not affected.
Encoding:	0 1 1 0 0 1 1 i	Example:	R0 contains the address 20H. The Accumulator holds the value 75H (01101010B). The instruction XCHD A,R0 will leave RAM location 20H holding the value 75H (01101010B) and 75H (01101010B) in the Accumulator.
Operation:	XRL (A) ← (A) ∨ ((Ri))		
XRL A,#data		XCHD A,R0	
Bytes:	2	Function:	Exchange Digit
Cycles:	1	Description:	XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal register specified by the instruction. The high-order nibble (bits 7-4) is not affected.
Encoding:	0 1 1 0 0 1 0 0	Example:	R0 contains the address 20H. The Accumulator holds the value 75H (01101010B). The instruction XCHD A,R0 will leave RAM location 20H holding the value 75H (01101010B) and 75H (01101010B) in the Accumulator.
		Bytes:	1
		Cycles:	1
Operation:	XRL (A) ← (A) ∨ #data	Encoding:	1 1 0 1 0 1 1 i
XRL direct,A		XCHD A,R0	
Bytes:	2	Function:	Exchange Digit
Cycles:	1	Description:	XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal register specified by the instruction. The high-order nibble (bits 7-4) is not affected.
Encoding:	0 1 1 0 0 0 1 0	Example:	R0 contains the address 20H. The Accumulator holds the value 75H (01101010B). The instruction XCHD A,R0 will leave RAM location 20H holding the value 75H (01101010B) and 75H (01101010B) in the Accumulator.
		Bytes:	1
		Cycles:	1
Operation:	XRL (direct) ← (direct) ∨ (A)	Encoding:	1 1 0 1 0 1 1 i



XRL direct, # data

Bytes: 3

Cycles: 2

Encoding:

0	1	1	0
0	0	1	1

direct address

immediate data

Operation:

XRL

(direct) \leftarrow (direct) \vee # data

Architectural Overview of the 13

87C51GB, 83C51GB

& 80C51GB

Architectural Overview of the
87C51GB, 83C51GB
& 80C51GB

13

87C51GB/83C51GB/80C51GB CHMOS SINGLE-CHIP 8-BIT MICROCONTROLLER ARCHITECTURAL OVERVIEW

AUTOMOTIVE

1.0 FUNCTIONAL DESCRIPTION

The 80C51GB is a highly integrated 8-bit microcontroller based on the MCS®-51 architecture. Its key features include a Serial Expansion Port, an 8-bit A/D converter, a flexible timer/counter subsystem, hardware support for operation in electrically noisy environments, and security for the on-chip EPROM memory. As a member of the MCS-51 family of devices, the 80C51GB is optimized for control applications. Its architecture and instruction set facilitate efficient utilization of the on-chip memory and peripheral resources.

The 80C51GB is a superset of the 80C51 microcontroller. It is software compatible with the 80C51 allowing use of existing software development tools.

The 80C51GB is ROMless, 83C51GB is the mask ROM, and the 87C51GB is the EPROM version. 80C51 will be used in this document.

1.1 80C51GB Features

The features of the 80C51GB are:

- 8 Channel, 8-Bit A/D Converter
- Two 5 Channel Programmable Counter Arrays with Independent
 - Input Capture
 - Output Compare
 - Pulse Width Modulation
- 3 16-Bit Timer/Counters
- Dedicated Watchdog Timer
- Oscillator Failure Detection
- 6 8-Bit Parallel I/O Ports with Schmitt Trigger Inputs
- Enhanced Serial I/O Port
- Serial Expansion Port
- 8K Byte ROM/EPROM
- 256 Byte RAM
- Security for EPROM
- Power Saving Standby Modes
- Choice of 68 Pin PLCC or CERQUAD Packages

Figure 1 shows a block diagram of the 80C51GB.

1.2 Compatibility with MCS®-51 Family

The 80C51GB is compatible with the 87C51 and 87C51FA as shown in Table 1.

Table 1.

80C51GB Feature	87C51	87C51FA
Instruction Set	Same	
Special Function Registers	Compatible	
Port 0, 1, 2, 3	Same	Same
Serial Port	Compatible	Same
Timer 0, 1	Same	Same
Timer 2	N/A	Same
Interrupts	Compatible	Compatible
Programmable Counter Array	N/A	Same
EPROM	Compatible	Compatible
RAM	Compatible	Same

DEFINITIONS:

N/A: This 80C51GB feature is not present in the 87C51 or 87C51FA.

SAME: This 80C51GB feature is identical to the corresponding 87C51 or 87C51FA feature.

COMPATIBLE: This 80C51GB feature is improved relative to the 87C51 or 87C51FA, but is compatible if the improvement is not utilized.

INCOMPATIBLE: This 80C51GB feature is not compatible to the corresponding 87C51 or 87C51FA feature.

2.0 FEATURE DESCRIPTIONS

2.1 EPROM, RAM and Memory Security

The 87C51GB contains 8K bytes of EPROM and 256 bytes of RAM. In addition, the 80C51GB contains a security feature which protects the on-chip EPROM from external access, preventing unauthorized read out of the EPROM. The MCS-51 architecture has separate address spaces for Program Memory, Data Memory and Internal RAM. The Program and Data Memory spaces are each 64K bytes long. The Internal RAM space is 256 bytes.

EPROM

The 8K EPROM occupies address locations 0000H to 1FFFFH in the program memory space. The EPROM also contains 64 bytes of key EPROM.

RAM

The full 256 bytes allowed by the architecture are implemented. The internal RAM is used for data regis-

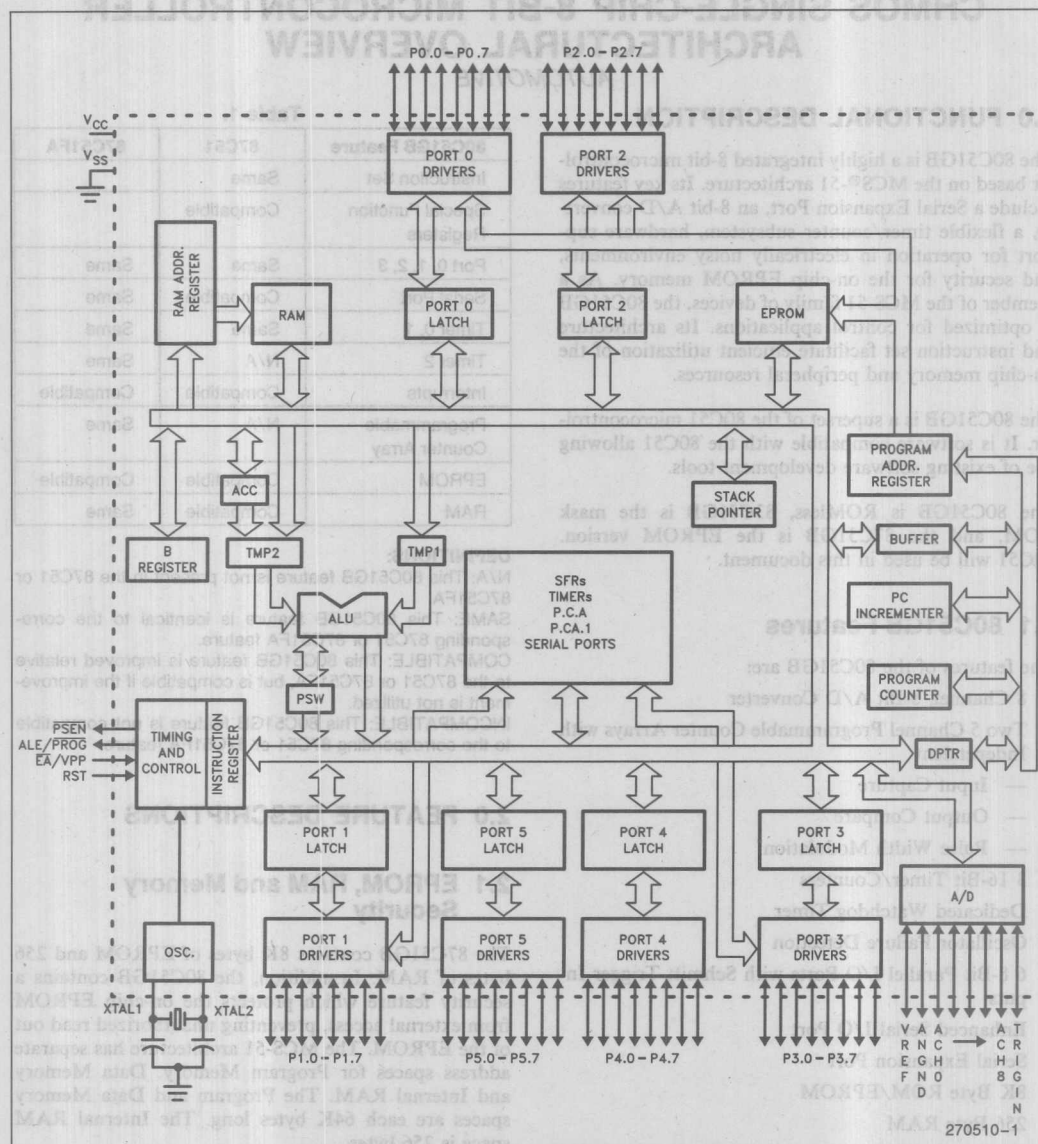


Figure 1. 87C51GB Block Diagram

ters, the address stack, and other data storage requirements.

MEMORY SECURITY

The 87C51GB contains two security features which protect the code of the on-chip EPROM. The key EPROM security encrypts the code during memory

verify and the two Security Bits can deny access to the EPROM array.

There are 64 bytes of key in the array that are used to encrypt the code during memory verify. Every time a code byte is addressed, bits PAR0-PAR4 are used to address a key byte. The data that is output on Port 0 is an XOR of the key byte and the code. The key bytes cannot be verified. Therefore, only the user who programs the key bytes would be able to decrypt the

scrambled code from the verify output. If the key array is not programmed, all key bytes read FFH and the encrypted data is equal to the program data.

Two security bits can be programmed by the user. These bits can provide additional code access protection. There are four ways that the EPROM can be accessed for a read:

- 1) A MOV C instruction in the external program memory can access internal codes.
- 2) Toggling EA pin.
- 3) Programming the EPROM array (e.g. programming all key bytes with 00H)
- 4) Code Verify Mode.

It is possible to read internal code by executing externally and periodically toggling the EA pin. A MOV A, instruction in the external program memory would move internal program memory to the accumulator if EA is initially low and brought high before State 4 of the MOV A, instruction. This ensures that the opcode of MOV A, is read from external memory while the data is fetched from the internal memory. The 80C51GB's EA pin is latched by the falling edge of POC (internal reset). This prohibits the ability of toggling EA to get the EPROM program to dump out of the ports by toggling EA.

The internal code would be most secure if the code verify is completely disabled. although programming the key bytes provides an encrypted verify, the code becomes accessible if the key bytes are known.

Security Bit 1 provides external execution security by disabling EPROM read modes 1, 2, and 3 mentioned above. Security bit 2 provides the security that Bit 1 offers and it also disables code verify completely. Table 2 lists the security types activated by different bits conditions.

2.2 A/D Converter

The 80C51GB A/D converter will have a resolution of 8 bits and an accuracy of $\pm 1/2$ LSB. The conversion

time for a single channel will be 20 μ sec at a clock frequency of 16 MHz with the sample and hold function included. Independent supply voltages are provided for the A/D. Also, the A/D will operate both in normal mode or in idle mode.

The A/D has 8 analog input pins; ACH0 (A/D Channel 0) ACH7, 1 reference input pin; COMPREF (COMParison REFerence), 1 control input pin; TRIGIN (TRIGger IN), and 2 power pins; VREF (Voltage REFerence) and ANGND (ANalog GrouND). In addition, the A/D has 8 conversion result registers; AD0 (A/D result for channel 0) AD7, 1 comparison result register; ACMP (Analog CoMParison), and 1 control register; ACON (A/D Control).

The control bit ACE (A/D Conversion Enable) in ACON controls whether the A/D is in operation or not. ACE=0 idles the A/D. ACE=1 enables A/D conversion. The control bit AIM (A/D Input Mode) in ACON controls the mode of channel selection. AIM = 0 is the scan mode, and AIM = 1 is the select mode. The result registers AD4 .. AD7 always contain the result of a conversion from the corresponding channels ACH4 .. ACH7. However, the result registers AD0 .. AD3 depend on this mode. In the scan mode, AD0 .. AD3 contain the values from ACH0 .. ACH3. In the select mode, one of the four channels ACH0 .. ACH3 is converted four times, and the four values are stored sequentially in locations AD0 .. AD3. Its channel is selected by bits ACS1 and ACS0 (A/D Channel Select 1 and 0) in ACON.

The control bit ATM (A/D Trigger Mode) in ACON controls the mode of execution. ATM = 0 is continuous mode, AT = 1 is trigger mode. In continuous mode conversions from channel 0 to 7 are repeated while ACE bit is set. In trigger mode, the conversion is triggered on the trailing edge of pin TRIGIN, and executed only once from channel 0 to 7. Any further trailing edge detection of TRIGIN will be ignored until the current conversion cycle has completed. The AIF (A/D Interrupt Flag) in ACON is set when a A/D conversion cycle from channel 0 to 7 has been completed, i.e. when AD7 is written. The EAD (interrupt Enable for A/D) in IEA (Interrupt Enable Register) de-

Table 2.

Bit 2	Bit 1	Security
Unprogrammed	Unprogrammed	None. Modes 1-4 of code access are all possible.
Unprogrammed	Programmed	Modes 1-3 are disabled
Programmed	Programmed	Modes 1-4 are disabled
Programmed	Unprogrammed	Invalid combination

termines whether an interrupt will be generated or not. If EAD = 1, an interrupt to the CPU occurs after the A/D completed conversion. All channels are also automatically compared against a single voltage reference provided by pin COMPREF. If the particular channel voltage is greater than the reference, the corresponding

bit in register ACMP is set. If not, the bit is reset. Notice that this comparison is done in channel sequence regardless of the mode being selected, that is, the A/D could be programmed in the select mode, but ACMP would still store the comparison result between AN0 - AN7 and COMPREF.

Table 3. The Mode of The Execution and Channel Selection

[Mode of Channel Selection]			Comparison Result	All Mode
Conversion Result	Scan Mode	Select Mode		
AD0	← ACH0	← ACHx	CMP0	← ACH0
AD1	← ACH1	← ACHx	CMP1	← ACH1
AD2	← ACH2	← ACHx	CMP2	← ACH2
AD3	← ACH3	← ACHx	CMP3	← ACH3
AD4	← ACH4	← ACH4	CMP4	← ACH4
AD5	← ACH5	← ACH5	CMP5	← ACH5
AD6	← ACH6	← ACH6	CMP6	← ACH6
AD7	← ACH7	← ACH7	CMP7	← ACH7
(if ACS1,0 = 00 then x = 0, ACS1,0 = 01 then x = 1, ACS1,0 = 10 then x = 2, ACS1,0 = 11 then x = 3)				

A/D Special Function Register

Table 4. Bit Assignments

Name	Bit								Address
	7	6	5	4	3	2	1	0	
ACON	—	—	AIF	ACE	ACS1	ACS0	AIM	ATM	97H
ACMP	CMP0	CMP1	CMP2	CMP3	CMP4	CMP5	CMP6	CMP7	C7H
AD0	MSB	...	← 8-bit A/D result for ACH0	→	LSB	84H
AD1	MSB	...	← 8-bit A/D result for ACH1	→	LSB	94H
AD2	MSB	...	← 8-bit A/D result for ACH2	→	LSB	A4H
AD3	MSB	...	← 8-bit A/D result for ACH3	→	LSB	B4H
AD4	MSB	...	← 8-bit A/D result for ACH4	→	LSB	C4H
AD5	MSB	...	← 8-bit A/D result for ACH5	→	LSB	D4H
AD6	MSB	...	← 8-bit A/D result for ACH6	→	LSB	E4H
AD7	MSB	...	← 8-bit A/D result for ACH7	→	LSB	F4H

Table 5

Bit 1	Bit 2
Unprogrammed	Unprogrammed
Unprogrammed	Unprogrammed
Programmed	Programmed
Programmed	Programmed
Unprogrammed	Unprogrammed

Table 5. Bit Descriptions

AIF	A/D Interrupt Flag. This bit is set when an A/D conversion cycle has been completed.
ACE	A/D Conversion Enable. This bit, when set by software, enables the A/D converter.
ACS1-0	A/D Channel Select. These two bits specify an A/D channel for the Select mode.
	ACS1 ACS0 Selected Channel
	0 0 ACH0
	0 1 ACH1
	1 0 ACH2
	1 1 ACH3
AIM	A/D Input Mode. This bit selects between the Scan and Select modes for the A/D converter. When AIM is low, the Scan mode is enabled.
ATM	A/D Trigger Mode. This bit selects between Continuous and Triggered conversion modes. When ATM is set high by software, the Triggered conversion mode is enabled. When ATM is low, the Continuous mode is enabled.
CMP0-CMP7	Comparison results with pin COMPREF. Pins ACH0-7 are automatically compared against a single voltage reference provided by pin COMPREF. If pin ACH0-7 is greater than the reference, CMP0-7 are set high. If not, CMP0-7 are set low. See Interrupts section for the following bits.
EAD	Interrupt Enable for A/D. This bit is in IEA register.
PAD	Interrupt Priority for A/D. This bit is in IPA register.
PAD.1	Interrupt Priority for A/D. This bit is in IPA1 register.

2.3 Programmable Counter Arrays

The Programmable Counter Arrays (PCA-PCA1) are each made up of a Counter Module and five Register/

Comparator Modules as shown below. The 16-bit output of the counter module is available to all five Register/Comparator Modules, providing one common timing reference. Each Register/Comparator Module is associated with a pin of Port 1/Port 4 and is capable of performing input capture, output compare and pulse width modulation functions. The PCAs are exactly the same in function except for the addition of clock input sources on PCA1.

The PCA Counter and five Register/Comparator Modules each have a status bit in the CCON/C1CON Special Function Registers. These six status bits are set according to the selected modes of operation described below. The CCON/C1CON Register provides a convenient means to determine which of the six PCA interrupts has occurred. The EC Bit in the IE (Interrupt Enable) Special Function Register is a global interrupt enable for the PCA.

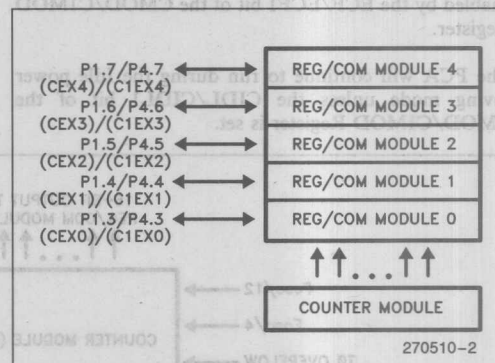


Figure 2. Programmable Counter Arrays

PCA COUNTER MODULES

The PCA counter is incremented in response to a count pulse from one of four input sources. As shown below, the allowed input sources are: the crystal oscillator through a prescaler of either modulus 12 or modulus 4, the Timer 0 overflow, and the P1.2 pin of Port 1. Incrementing of the counter is enabled by the CR bit of the CCon Special Function Register.

The PCA1 counter is incremented in response to a count pulse from one of seven input sources. As shown below, the allowed input sources are: the crystal oscillator through a prescaler of either modulus 12 or modulus 4, the Timer 0 overflow, the P1.2 pin of Port 1, the Timer 1 overflow, the P4.2 pin of Port 4, and the T2/P1.7 of Port 1. Incrementing of the counter is enabled by the CR1 bit of the C1CON Special Function Register. This allows the user to configure the two PCA modules to run off the same timing source if one of the first four input sources is chosen for PCA1.

timers as described, the timers can also be enabled simultaneously. This can be achieved by setting the CRE bit of the C1CON Special Function Register. When CRE is set, PCA1 counter is enabled whenever both the CR and CR1 bits are set, and is disabled when any one of them is cleared. Thus the user can first set the CR1 bit, and then can start and stop both the counters simultaneously by setting and clearing CR bit respectively.

The counter can be accessed by the CPU through the CH/CH1 (Counter High) and CL/CL1 (Counter Low) Special Function Registers. Reading is allowed at any time. However, writing to the PCA counters is inhibited while the counters are running.

Overflow of the PCA counter sets the CF/CF1 flag in the CCON/C1CON Register, and causes an interrupt if enabled by the ECF/ECF1 bit of the CMOD/C1MOD Register.

The PCA will continue to run during the Idle power saving mode unless the CIDL/CIDL1 bit of the CMOD/C1MOD Register is set.

The PCA has five Register/Comparator Modules. Each Register/Comparator Module can be independently configured to provide Input Capture, Output Compare, or Pulse Width Modulation (PWM) functions. The basic structure is shown below.

The Compare/Capture Registers are accessed through the CCAPnH/C1CAPnH and CCAPnL/C1CAPnL Special Function Registers. They can be read or written at any time. The five CCAPMn/C1CAPMn Special Function Registers select the modes of operation for their corresponding Register/Comparator Modules.

In the Input Capture mode a 1-to-0 or 0-to-1 transition at the CEXn/C1EXn pin generates a capture strobe which sets the status bit CCFn/C1CFn and stores the current value of the PCA Counter into the nth Compare/Capture Register. This mode allows the time when an external event occurs to be recorded with a resolution equal to the period of the Counter Module input clock. In this mode, the accuracy of timing measurement is not affected by interrupt latency.

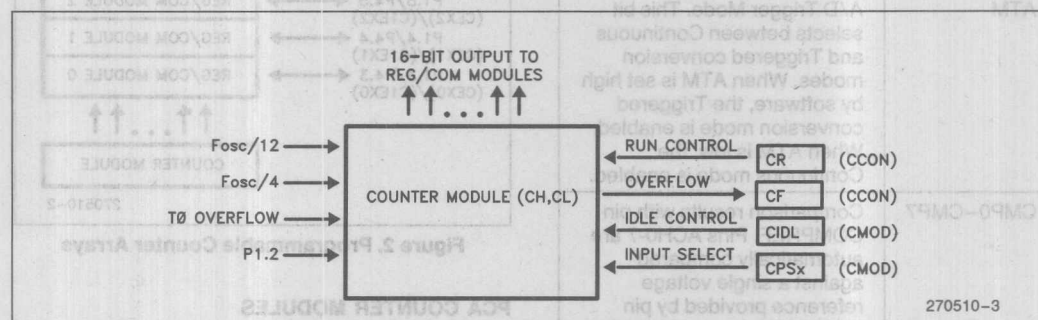


Figure 3. PCA Counter Module

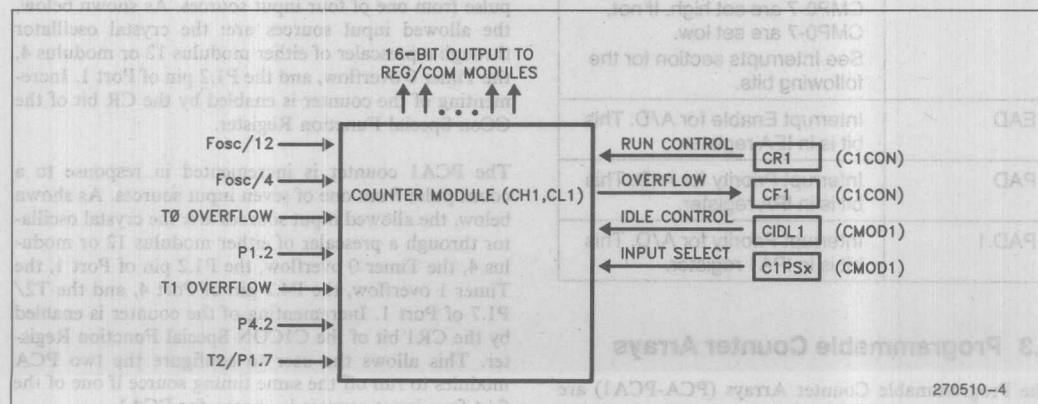


Figure 4. PCA1 Counter Module

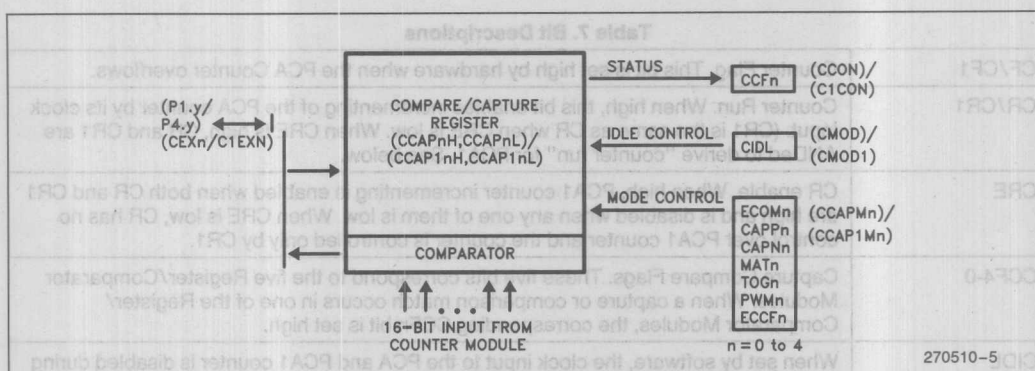


Figure 5. PCA Register/Comparator Module

In the Output Compare mode the comparator continuously compares the contents of the Compare/Capture register with the PCA Counter. When the register contents equal the PCA Counter, the CCFn/C1CFn bit is set and, if the TOGn/TOG1n bit is set high, the corresponding CEXn output pin is toggled. If TOGn is set low, then the CEXn pin is unaffected, providing a software timer without tying up a port pin. The Output Compare mode allows the timing of output transitions with a resolution equal to the period of the Counter Module input clock. In addition, accurate delays between input events and output events can be set up using the Input Capture mode for the input event and the Output Compare mode for output.

In the PWM mode the comparator continuously compares the low byte of the Compare/Capture register with the low byte of the PCA Counter. When the low byte of the register equals the low byte of the counter, the CEXn/C1EXn output pin is set high. When the low byte of the counter overflows, the output pin is set low. The duty cycle of the output waveform is directly proportional to the value contained in the Compare/Capture Register. The duty cycle may be changed at any time by writing to the high byte of the Compare/Capture Register. In the PWM mode, the high byte of the

Compare/Capture Register serves as a buffer to provide glitch-free PWM waveform generation. The frequency of the output waveform is the input clock rate of the PCA Counter divided by 256.

PCA WATCHDOG TIMER

In addition to the hardware Watchdog Timer that exists on the 80C51GB, there is a programmable frequency Watchdog Timer in the PCA that operates off of Module 4. When Module 4 is in Watchdog Timer mode, every time the count in the PCA matches the value stored in compare/capture module 4, a reset will internally occur. This reset does NOT drive the user RESET pin. The PCA Module 4 enters this mode when the WDTE bit is set in the CMOD register of PCA. This mode is not implemented on PCA1.

With this additional Watchdog Timer the user can customize the number of machine cycles that should elapse before his software will clear the Watchdog Timer in the PCA. This would be done if the user desired fewer possible machine cycles that the part could execute in a software run away mode, before the Watchdog Timer would reset the part.

PCA SPECIAL FUNCTION REGISTERS

Table 6. Bit Assignments

Name	Bit							
	MSB							LSB
	7	6	5	4	3	2	1	0
CCON	CF	CR	—	CCF4	CCF3	CCF3	CCF1	CCF0
CMOD	CIDL	WDTE	—	—	—	CPS1	CPS0	ECF
CCAPMn	—	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn
C1CON	CF1	CR1	CRE	C1CF4	C1CF3	C1CF3	C1CF1	C1CF0
C1MOD	CIDL1	—	—	—	C1PS2	C1PS1	C1PS0	ECF1
C1CAPMn	—	E1COMn	CAP1Pn	CAP1Nn	MAT1n	TOG1n	PWM1n	E1CCFn

Table 7. Bit Descriptions

CF/CF1	Counter Flag. This bit is set high by hardware when the PCA Counter overflows.			
CR/CR1	Counter Run. When high, this bit enables incrementing of the PCA counter by its clock input. (CR1 is the same as CR when CRE is low. When CRE is high, CR and CR1 are ANDed to derive "counter run" for PCA1. See below.			
CRE	CR enable. When high, PCA1 counter incrementing is enabled when both CR and CR1 are high and is disabled when any one of them is low. When CRE is low, CR has no control over PCA1 counter and the counter is controlled only by CR1.			
CCF4-0	Capture Compare Flags. These five bits correspond to the five Register/Comparator Modules. When a capture or comparison match occurs in one of the Register/Comparator Modules, the corresponding CCFn bit is set high.			
CIDL	When set by software, the clock input to the PCA and PCA1 counter is disabled during the Idle power reduction mode. When this bit is low, the PCAs operate during Idle.			
WDTE	PCA Watchdog Timer enable bit. When set, a match signal of Register/Comparator Module 4 causes an internal reset. In order to use Register/Comparator Module 4 as a watchdog timer the operating mode should be 16-bit software timer mode or high speed output mode. When WDTE is clear Register/Comparator Module 4 of PCA does not effect the Reset of the 80C51GB.			
CPS1-2	Counter Pulse Select. These bits select the clock input for the PCA counter.			
		CPS1	CPS0	Selected PCA Input
		0	0	Crystal oscillator divided by 12 (Fosc/12)
		0	1	Crystal oscillator divided by 4 (Fosc/4)
		1	0	Timer 0 overflow
		1	1	External Clock input on P1.2 pin (Fosc/8 max.)
	CPS2	CPS1	CPS0	Selected PCA1 Input
	0	0	0	Crystal oscillator divided by 12 (Fosc/12)
	0	0	1	Crystal oscillator divided by 4 (Fosc/4)
	0	1	0	Timer 0 overflow
	0	1	1	External clock input on P1.2 pin (Fosc/8 max.)
	1	0	0	Same as 000, not to be used.
	1	0	1	External clock input on T2/P1.0 pin (Fosc/8 max.)
	1	1	0	Timer 1 overflow
	1	1	1	External clock input on P4.2 pin (Fosc/8 max.)
ECF/ECF1	This bit enables an interrupt on counter overflow. This bit does not affect the CF status bit in the PCA Control Register.			
ECOMn/E1COMn	Enable Comparator. This bit when set high enables the comparator of the nth Register/Comparator Module. It is automatically reset when the CCAPnL register is written and set when the CCAPnH register is written, thereby suppressing unwanted comparator matches while the 16-bit Compare/Capture Register is being loaded.			
CAPPn/C1APPn	Capture Positive. This bit when set high enables Input Capture by a positive transition on the nth CEX pin. When this bit is low Input Capture by the positive transition is disabled.			
CAPNn/CAP1Nn	Capture Negative. This is identical to CAPPn except that it enables Input Capture on negative transitions. Both CAPPn and CAPNn may be set to enable Input capture on both positive and negative transitions.			
MATn/MAT1n	Match. This bit when set high enables setting the nth CCF bit by a match in the corresponding Comparator. When this bit is set low the CCF bit is not set by a match.			

Table 7. Bit Descriptions (Continued)

TOGn/TOG1n	Toggle. This bit when set high enables toggling the nth CEX pin by a match in the corresponding Comparator. When this bit is set low the CEX pin is not toggled by a match.					
PWMn/PWM1n	Pulse Width Modulation mode. When set high, this bit configures the nth Register/Comparator Module for the PWM mode.					
ECCFn/E1CCFn	Enable CCF interrupt. When set high, this bit enables the CCF bit to generate an interrupt request. The ECCF bit has no affect on the CCF bit itself. The CCAPMn/CCAP1Mn registers should be programmed as follows to configure the Register/Comparator Module.					
	C	C	M	T	P	E
	A	A	A	O	W	C
	P	P	T	G	M	C
	P	N	n	n	n	F
	n	n				n
	Operating Mode (Same for PCA1)					
	0	0	0	0	0	0
	1	0	0	0	0	X
	0	1	0	0	0	X
	1	1	0	0	0	X
	0	0	1	0	0	X
	0	0	1	1	0	X
	0	0	0	0	1	0
	No operation					
	Input Capture with positive edge					
	Input Capture with negative edge					
	Input Capture with both edges					
	Software Timer					
	Output Compare					
	PWM					

2.4 Timer/Counters

The 80C51GB contains three general purposes, 16-bit timer/counters. Each can be configured to operate in a variety of modes as described below.

Timer 0 - Timer 0 can be configured to operate as either a timer or an event counter. The timer or counter function is selected by the C/T0 bit of the TMOD Special Function Register. When C/T0 is low, Timer 0 is incremented every machine cycle. Since a machine cycle consists of 12 oscillator clocks, the clock rate is 1/12 the crystal oscillator frequency. When C/T0 is high, Timer 0 is incremented by 1-to-0 transitions at the T0 pin. Incrementing of Timer 2 is enabled by the TR0 bit of the TCON Register, the GATE0 bit of the TMOD Register, and the INT0 pin. When the GATE bit is low, Timer 0 is enabled solely by the TR0 bit. When the GATE bit is high, Timer 0 is enabled by the TR0 bit and gated by the INT0 pin. When Timer 0 overflows, it sets the TF0 bit of the TCON Register, and generates an interrupt request if enabled by the ET0 bit of the IE Special Function Register. The 16-bit Timer 0 Register is accessible through the TH0 and TL0 Registers. The high byte is accessed through TH0, and the low byte through TL0. The M10 and M20 bits of the TMOD Register select one of four operating modes for Timer 0.

Timer 1 - Timer 1 is identical to Timer 0 except that in Mode 3, Timer 1 simply holds its count value. The effect is the same as setting the TR1 bit low. In addition,

Table 8. Timer/Counter Modes

Mode 0	Timer 0 is configured as an 8-bit timer (TL0) with a module 32 prescaler (TH0).
Mode 1	Timer 0 is configured as a 16-bit timer.
Mode 2	Timer 0 is configured as an 8-bit timer (TL0) with automatic reload from TH0 on overflow.
Mode 3	Timer 0 is configured as two 8-bit timers with TL0 controlled by the Timer 0 control bits, and TH0 controlled by the Timer 1 control bits. This mode is useful when Timer 1 is used as the Serial Port baud rate generator.

tion, Timer 1 can be selected to serve as the baud rate generator for the serial port. Mode 2 is most useful for this purpose. Timer 1 has its own independent set of controls and registers (ie. TR1, TR2, GATE1, C/T1, M11, M01, TH1, TL1, T1, INT1, and ET1).

Timer 2 - Like Timer 0 and Timer 1, Timer 2 has several programmable modes. The C/T2, TR2, TF2, and ET2 control bits are similar to the corresponding bits in the other two timers. Timer 2 has three operating modes: Capture, Auto-Reload, and Baud Rate Generator. The CP/RL2, RCLK, and TCLK bits in the TCON Register select the operating mode. Timer 2 may be selected to serve as the baud rate generator for the serial port.

Timer 2 Capture Mode - In the Capture mode, Timer 2 operates as an upcounter with a capture strobe. A 1-to-0 transition on the T2EX pin causes the current value of Timer 2 (T2H, T2L) to be captured into the Reload/Capture Registers (RCAP2H, RCAP2L). In addition, the transition at T2EX sets the EXF2 bit, and overflow of Timer 2 sets the TF2 bit. Either bit can cause an interrupt request if enabled by ET2.

Timer 2 Auto-Reload Mode - In the Auto-reload mode, Timer 2 can be configured as an upcounter with a reload strobe, or as an up/down counter. The DCEN bit selects between the two modes.

When DCEN is low, Timer 2 is configured as an upcounter which is reloaded with the value of the Reload/Capture Register (RCAP2H, RCAP2L) when Timer 2 overflows or when a 1-to-0 transition occurs on the T2EX pin (if enabled by EXEN). As in the Capture mode, the transition at T2EX sets the EXF2 bit and overflow of Timer 2 sets the TF2 bit. Either bit can cause an interrupt request if enabled by ET2.

When DCEN is high, Timer 2 is configured as an up/down counter, with T2EX determining the counting

direction. When T2EX is high Timer 2 counts up and is reloaded with the contents of the Reload/Capture register upon overflow (ie. when Timer 2 = FFFFH). When T2EX is low Timer 2 counts down and is reloaded with FFFFH upon overflow (ie. when Timer 2 = Reload/Capture). Overflow of Timer 2 sets the TF2 bit and inverts the EXF2 bit. Either bit can cause an interrupt request if enabled by ET2.

Timer 2 Baud Rate Generator Mode - The Baud Rate Generator mode is similar to the Auto-Reload mode in that Timer 2 is configured as an upcounter and is reloaded on overflow. However when C/T2 is low, Timer 2 is incremented at $\frac{1}{2}$ the oscillator frequency instead of 1/12 as in the other modes. In this mode overflow does not set the TF2 bit and does not generate interrupt requests. If EXEN2 is set, a 1-to-0 transition on the T2EX pin will set the EXF2 bit and can generate an interrupt request. However, in this mode T2EX will not cause a reload. Thus when Timer 2 is used as a baud rate generator, T2EX can be used as an external interrupt. In this mode, the T2H, T2L, RCAP2H, and RCAP2L Registers should not be read or written while Timer 2 is running. They can be accessed if Timer 2 has been stopped by setting the TR2 bit low.

TIMER/COUNTER SPECIAL FUNCTION REGISTERS

Table 9. Bit Assignments

Name	Bit							
	MSB							LSB
	7	6	5	4	3	2	1	0
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
TMOD	GATE1	C/T1	M11	M01	GATE0	C/T0	M10	M00
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
T2MOD	—	—	—	—	—	—	—	DCEN
IE	EA	EC	ET2	ES	ET1	EX1	ET0	EX0

Table 10. Bit Descriptions

TF1	Timer 1 overflow Flag. This bit is set by hardware when Timer 1 overflows. It is cleared by hardware when the processor vectors to the Timer 1 interrupt routine.
TR1	Timer 1 Run control bit. This bit is set or cleared by software to control incrementing of Timer 1. When TR1 is low, incrementing of Timer 1 is disabled.
TF0	Timer 0 overflow Flag. This bit is set by hardware when Timer 0 overflows. It is cleared by hardware when the processor vectors to the Timer 0 interrupt routine.
TR0	Timer 0 Run control bit. This bit is set or cleared by software to control incrementing of Timer 0. When TR0 is low, incrementing of Timer 0 is disabled.
IE1	See INTERRUPTS section.
IT1	See INTERRUPTS section.

IE0	See INTERRUPTS section.		
IT0	See INTERRUPTS section.		
GATE1	Gating control bit. When set by software incrementing of Timer is gated by the INT1 pin. When GATE1 is low INT1 does not affect operation of Timer 1.		
C/T1	Counter or Timer select bit. When set by software Timer 1 is configured as an event counter (incremented by 1-to-0 transitions on the T1 pin). When C/T1 is low, Timer 1 is configured as a timer (incremented once each machine cycle).		
M11-01	Mode control bits. These two bits are set by software to select the mode of operation for Timer 1.		
	M11	M01	Timer 1 Mode
	0	0	MODE 0
	0	1	MODE 1
	1	0	MODE 2
	1	1	MODE 3
GATE0	Gating control bit. When set by software incrementing of Timer is gated by the INT0 pin. When GATE0 is low INT0 does not affect operation of Timer 0.		
C/T0	Counter or Timer select bit. When set by software Timer 0 is configured as an event counter (incremented by 1-to-0 transitions on the T0 pin). When C/T0 is low, Timer 0 is configured as a timer (incremented once each machine cycle).		
M10-00	Mode control bits. These two bits are set by software to select the mode of operation for Timer 0.		
	M10	M00	Timer 0 Mode
	0	0	MODE 0
	0	1	MODE 1
	1	0	MODE 2
	1	1	MODE 3
TF2	Timer 2 overflow flag. This bit is set by hardware when Timer 2 overflows in the Capture or Auto-Reload modes. This bit is not set in the Baud Rate Generator mode. It can initiate an interrupt request if enabled. Since TF2 is not cleared by hardware, it should be cleared by software in the Timer 2 interrupt routine.		
EXF2	Timer 2 External Flag. This bit is set by hardware when a capture or reload is caused by the T2EX pin. It can initiate an interrupt request if enabled. EXF2 is not cleared by hardware, so it should be cleared in the Timer 2 interrupt routine.		
RCLK	Receive Clock bit. This bit is set by software to allow Timer 2 to serve as the baud rate generator for the Serial Port Receiver, and configures Timer 2 for the Baud Rate Generator mode.		
TCLK	Transmit Clock bit. This bit is set by software to allow Timer 2 to serve as the baud rate generator for the Serial Port Transmitter, and configures Timer 2 for the Baud Rate Generator mode.		
EXEN2	Timer 2 External Enable. This bit is set by software to enable T2EX to cause a capture or reload of Timer 2. When EXEN2 is low T2EX has no effect on Timer 2.		
TR2	Timer 2 Run control bit. This bit is set or cleared by software to control incrementing of Timer 2. When TR2 is low, incrementing of Timer 2 is disabled.		

Table 10. Bit Descriptions (Continued)

C/T2	Counter or Timer select bit. When this bit is set by software Timer 2 is configured as an event counter (incremented by 1-to-0 transitions on the T2 pin). When C/T2 is low, Timer 2 is configured as a Timer, incremented once each machine cycle in Capture and Auto-Reload modes, or incremented at 1/2 the crystal frequency in the Baud Rate Generator mode.			
CP/RL2	Capture/Reload control bit. When RCLK and TCLK are both low, CP/RL2 selects between the Capture and Auto-Reload modes. When CP/RL2 is high, the Capture mode is selected.			
DCEN	Down Count Enable. This bit is set by software to enable up/down counting in the Auto-reload mode.			
	RCLK	TCLK	CR/RL2	Timer 2 Mode
	0	0	0	Auto-Reload Mode
	0	0	1	Capture Mode
	0	1	X	Baud Rate Generator Mode
	1	0	X	Baud Rate Generator Mode
	1	1	X	Baud Rate Generator Mode
EA	See INTERRUPTS section.			
EC	See INTERRUPTS section.			
ES	See INTERRUPTS section.			
EX1	See INTERRUPTS section.			
EX0	See INTERRUPTS section.			
ET2	Enable Timer 2 interrupt. This bit is set by software to enable an interrupt request from Timer 2.			
ET1	Enable Timer 1 interrupt. This bit is set by software to enable an interrupt request from Timer 1.			
ET0	Enable Timer 0 interrupt. This bit is set by software to enable an interrupt request from Timer 0.			

2.5 Watchdog Timer

The 80C51GB contains a dedicated Watchdog Timer to allow recovery from a software upset. The Watchdog Timer consists of a 14-bit counter which is cleared on Reset, and subsequently incremented every machine cycle. The counter may be reset by writing 1EH and E1H in sequence to the WDTRST Special Function Register. If the counter is not reset before it reaches 16383 (all 14 bits equal 1), the chip will be forced into a reset sequence by the Watchdog Timer. The Watchdog Timer will not drive the external reset pin.

The relation between the Watchdog Timer and Power Reduction Modes deserves some special attention. The Watchdog Timer continues to count while the part is in IDLE mode. This means that the user must dedicate some internal or external hardware to service the

Watchdog during Idle. One way to service the Watchdog Timer would be to use one of the hardware timer/counters on the 80C51GB. The user could reset the timer/counter and enable interrupts to occur on the overflow of the timer/counter. The interrupt service routine could then clear the Watchdog Timer, reload the timer/counter for the next service period of the Watchdog Timer, and then put the part back into Idle. These activities are cumbersome for the Idle mode user but it is necessary to retain a Watchdog Timer that has the least probability of failing by inadvertently getting disabled.

The Power Down mode stops all Phase clocks. This means that the Watchdog Timer will stop counting and hold its count during Power Down Mode. The Watchdog will resume counting from where it left off

The serial port supports one synchronous mode and three asynchronous modes. Mode selection is accomplished by the SM1 and SM0 bits of the SCON Special Function Register.

MODE 0 - Mode 0 is a half-duplex synchronous mode in which the serial port provides a shift clock as well as serial data. The shift clock is supplied on the P3.1/TXD pin, and serial data enters or exits through the P3.0/RXD. The 8 data bits are transmitted and received LSB first. The baud rate is fixed at 1/12 the oscillator frequency. Transmission is initiated when the SBUF register is written. Reception is initiated by setting the REN bit of the SCON register.

MODE 1 - Mode 1 is a full-duplex, asynchronous mode. The serial frame consists of 10 bits, a start bit, 8 data bits, and a stop bit. The P3.0/RXD pin is used for reception, and the P3.1/TXD pin is used for transmission. On receive, the stop bit goes into the RB8 bit of the SCON register. The baud rate is generated by the overflow of Timer 1 or Timer 2.

MODES 2 and 3 - These modes are full-duplex, asynchronous modes. The serial frame consists of 11 bits, a start bit, 8 data bits, a programmable 9th bit, and a stop bit. On transmit, the 9th data bit can be assigned by writing the TB8 bit in the SCON register. This bit can be used for parity or a command/data flag. On receive, the 9th bit goes into the RB8 bit in the SCON register. In mode 2, the baud rate is programmable to either $\frac{1}{32}$ or $\frac{1}{64}$ of the oscillator frequency. In mode 3, the baud rate is generated by the overflow of Timer 1 or Timer 2.

FRAMING ERROR DETECTION - Framing error detection is provided in the three asynchronous modes. Framing error detection is enabled by setting the SMOD0 bit in the PCON register. SMOD0 is cleared by reset to maintain compatibility with the 80C51. When framing error detection is enabled, the SM0 bit in the SCON register is replaced with the framing error status bit FE. After a frame has been received the stop bit is checked. If it is invalid (low), the FE bit is set. Otherwise, the FE bit retains its previous value.

MULTIPROCESSOR COMMUNICATION - The serial port supports configuration as a slave processor

SERIAL PORT SPECIAL FUNCTION REGISTERS

Table 13. Bit Assignments

Name	MSB						LSB			
SCON	FE	SM1	SM2	REN	TB8	RB8	TI	RI	(SMOD0 = 1)	
SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	(SMOD0 = 0)	
PCON	SMOD1	SMOD0	—	POF	GF1	GF0	PD	IDL		
IE	EA	—	ET2	ES	ET1	EX1	ET0	EX0		

in an environment where multiple slave processors share a single serial line. The serial port supports differentiation of data and command frames, and recognition of slave and broadcast addresses. The SM2 bit in the SCON register enables the receiver to ignore frames which have the 9th bit cleared (ie. data frames destined for another slave). The SM2 bit also enables the receiver to check frames which have the 9th bit set (ie. command frames which select the next destination slave or slaves) for an address match. When the received command frame matches the slave address or the broadcast address, the RI bit is set and an interrupt is requested if enabled by ES. The received frame may then be read from SBUF. The slave and broadcast addresses may contain don't care bits which are masked off during address comparisons. Don't-care bits in the slave address are defined by zeros in the SADEN register. The remaining bits of the slave address are defined by the SADDR register. Don't-care bits in the broadcast address are defined by zeros in the result of the logical OR of the SADEN and SADDR registers. The remaining bits of the broadcast address are logic ones.

EXAMPLE

SADDR contains 1010 1010

SADEN contains 1111 1100

1010 10XX becomes slave address

1111 111X becomes broadcast address

BAUD RATE GENERATION - In modes 1 and 3, the serial port baud rate is generated by the overflow of Timer 1 or Timer 2. Timer 1 is the default baud rate generator for both the transmitter and the receiver. However, Timer 2 can be selected as the transmitter baud rate generator, the receiver baud rate generator, or both. This selection is made by setting the RCLK and TCLK bits of the T2CON register as follows:

Table 12. Baud Rate Generator Selection

RCLK	TCLK	Receiver Baud Rate Generator	Transmitter Baud Rate Generator
0	0	Timer 1	Timer 1
0	1	Timer 1	Timer 2
1	0	Timer 2	Timer 1
1	1	Timer 2	Timer 2

Table 14. Bit Descriptions

FE	Framing Error bit. This bit is set by the receiver when an invalid stop bit is detected. The FE bit is not cleared by valid frames. It should be cleared by software. The SMOD1 bit in the PCON register must be set to enable access to the FE bit.		
SM0-1	Serial port Mode. These two bits are set by software to select the mode of operation for the serial port.		
	SM0	SM1	Serial Port Mode
	0	0	MODE 0
	0	1	MODE 1
	1	0	MODE 2
	1	1	MODE 3
SM2	Serial port Mode. This bit is set by software to enable the multiprocessor communication features. SM2 enables differentiation of data and command frames, and recognition of slave and broadcast addresses.		
REN	Receiver Enable. This bit is set by software to enable the serial port receiver.		
TB8	Transmit Bit 8. This bit is set by software to specify the 9th data bit that will be transmitted in Modes 2 and 3.		
RB8	Receiver Bit 8. This bit is loaded by the receiver with the 9th data bit received in Modes 2 and 3, or with the stop bit in Mode 1.		
TI	Transmitter Interrupt. This bit is set by the transmitter after the last data bit of a frame has been transmitted. It is not cleared by hardware.		
RI	Receiver Interrupt. This bit is set by the receiver after the last data bit of a frame has been received. It is not cleared by hardware.		
SMOD1	Serial port Mode. This bit is set by software to enable detection of framing errors.		
SMOD0	Serial port Mode. This bit is set by software to double the baud rate when Timer 1 is used as the baud rate generator.		
POF	See section 2.10 on Power Reduction Modes		
GF1	See section 2.10 on Power Reduction Modes		
GF2	See section 2.10 on Power Reduction Modes		
PD	See section 2.10 on Power Reduction Modes		
IDL	See section 2.10 on Power Reduction Modes		
EA	See section 2.9 on Interrupts		
ET2	See section 2.9 on Interrupts		
ES	Enable Serial port interrupt. This bit is set by software to enable serial port interrupt requests.		
ET1	See section 2.9 on Interrupts		
EX1	See section 2.9 on Interrupts		
ET0	See section 2.9 on Interrupts		
EX0	See section 2.9 on Interrupts		

The 80C51GB contains a half-duplex, synchronous serial interface protocol. The SEPIO/P4.1 and SEPCLK/P4.0 are the pins that are used to communicate read/write operations to peripheral devices.

Serial transmission is initiated by the writing to the SEPDAT register. Serial reception is initiated by setting the SPREN bit in the SEPCON register.

The SEP has a programmable clock output pin. The clock frequency, latching edge and clock idle state is programmable by the user.

Programming The SEP Clock

The clock frequency is programmed by SEPS1 and SEPS0 as shown below.

SEPS1	SEPS0	XTAL Divide By
0	0	12
0	1	24
1	0	48
1	1	96

The divisor yields clock frequencies of 1333, 666, 333 and 166KHz at 16MHz XTAL operation. The Clock phase select is used to select the sampling edge of the data and the output edge of data. When CLKPHA is zero the SEP changes data on the second phase of the clock and samples data on the first phase of the clock. The data shift and sample points are reversed for the SEP when CLKPHA is set to '1'.

ty. When set to '0 the quiescent state of the SEPCLK is an output low. When Set to '1 the quiescent state of the SEPCLK is an output high. By using the CLKP and CLKPHA bits the user can select one of four master clock definitions.

Write/Read Operations With The SEP

A write transaction is initiated by writing to the SEP data reg with the SEPREN bit set to '0. This will start the clock logic and begin the shifting out of the contents of the SEP data reg MSB first for 8 bits. A read to the SEP is done by simply setting the SEPREN bit in the SEPCON. This initializes the SEPCLK to begin shifting and start the loading of the SEP data reg.

Depending on the value of the SEPIE bit in the SEPCON register an interrupt can be forced at the end of a read/write cycle but the SEPIF bit will always be set in the SEPSTAT Register.

Serial Expansion Port Fault Modes

The SEP has two fault mode bits. The SEPFWR bit is set when a read/write transaction is attempted while a write transaction is in progress. The write transaction in progress will complete. The attempted read/write that caused the fault will be ignored by the SEP.

The SEPFRD bit is set when a read/write transaction is attempted while a read transaction is in progress. The read transaction in progress will complete. The attempted read/write that caused the fault will be ignored by the SEP.

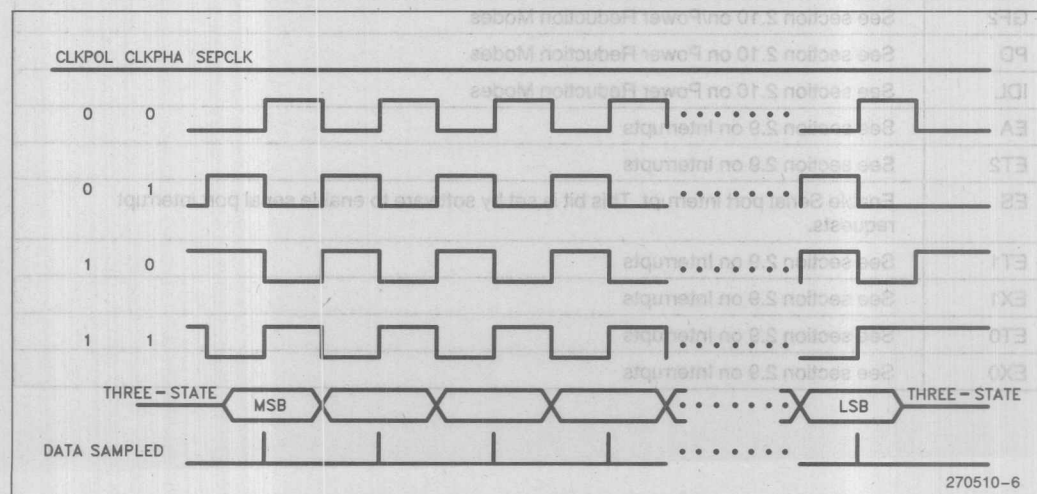


Figure 6. SEP Data Transaction

Table 14. Serial Expansion Port Special Function Registers

Name	Bit							
	MSB							LSB
	7	6	5	4	3	2	1	0
SEPSTAT	—	—	—			SEPFWR	SEPFRD	SEPIF
SEPCON	—	—	SEPE	SEPREN	CLKP	CLKPHA	SEPS1	SEPS0
SEPDATA	D7	D6	D5	D4	D3	D2	D1	D0

Table 15. Bit Descriptions

SEPFWR	The SEP write fault bit. This bit is set when the SEP is operated in a fault mode. The bit is set when the user attempts a read/write while a write transaction is ongoing. This bit must be cleared by software.
SEPFRD	The SEP read fault bit. This bit is set when the SEP is operated in a fault mode. The bit is set when the user attempts a read/write while a read transaction is ongoing. This bit must be cleared by software.
SEPIF	The SEP interrupt flag. This bit is cleared by reset and set after an SEP data transaction is complete. This bit must be cleared by software.
SEPE	The SEP enable bit. This bit when set enables the SEP data I/O and SEP clock signal to the external pins. This bit is cleared on reset.
SEPREN	The SEP receive enable. This bit, when set, starts the reception of data. In order to do a write this bit must be cleared. This bit is auto-cleared after 8 bits have been received by the SEP. This bit is cleared on reset.
CLKPOL	The SEP clock polarity select. When this bit is cleared and the SEP is in an idle mode the SEPCLK will remain at an output low. When this bit is set the SEPCLK will remain at an output high during idle. This bit is cleared on reset.
CLKPHA	The SEP clock phase select. This bit determines which clock edge will capture data and allow data to change. This bit is cleared on reset.
SEPS1	The SEP speed select bit1. This bit in combination with SEPS0 determines what frequency the SEP module will operate in.
SEPS0	The SEP speed select bit0. This bit in combination with SEPS1 determines what frequency the SEP module will operate in.
D0-D7	The SEP data bits. These are the SEP I/O data.

2.8 Parallel I/O Ports

The 80C51GB contains six 8-bit parallel I/O ports. All six ports are bidirectional and consist of a latch, an output driver, and an input buffer. Many of the port pins have multiplexed I/O and control functions.

PORT PINS AS OUTPUTS

Port 0 has open drain outputs when it is not serving as the external data bus. The internal pullup is active only when the pin is outputting logic 1 during external memory access. An external pullup resistor is required on Port 0 when it is serving as an output port.

Ports 1, 2, 3, 4 and 5 have quasi-bidirectional outputs. A strong pullup provides a fast rise time when the pin is set to a logic 1. This pullup turns on for two oscillator periods to drive the pin high and then turns off. The pin is held high by a weak pullup.

Writing the P0, P1, P2, P3, P4 or P5 Special Function Register sets the corresponding port pins. All six port registers are bit addressable.

PORT PINS AS INPUTS

The pins of all six ports are configured as inputs by writing a logic 1 to them. Since Port 0 is an open drain

port, it provides a very high input impedance. Since pins of Port 1, 2, 3, 4 and 5 have weak pullups (which are always on), they source a small current when driven low externally. All six ports have Schmitt trigger inputs with a minimum hysteresis of 0.4 volts.

PORT STATES DURING RESET

Ports 0, 1 and 3 reset asynchronously to a one and Ports 2, 4, and 5 reset to a zero asynchronously. Ports 2, 4 and 5 will reset to a one when the reset is done to activate a test mode.

ALTERNATE PORT FUNCTIONS

Ports 0, 1, 2, 3, 4 and 5 have alternate functions as well as their I/O function as described below.

Table 16. Alternate Port Functions

Port Pin	Alternate Function
P0.0/AD0 - P0.7/AD7	Multiplexed Address/Data for External Memory
P1.0/T2	Timer 2 External Clock Input
P1.1/T2EX	Timer 2 Reload/Capture/Direction Control
P1.2/C	PCA External Clock Input
P1.3/CEX0 - P1.7/CEX4	PCA Capture Input, Compare/PWM Output
P2.0/A8 - P2.7/A15	High Byte of Address for External Memory
P3.0/RXD	Serial Port Input
P3.1/TXD	Serial Port Output
P3.2/INT0	External Interrupt
P3.3/INT1	External Interrupt
P3.4/T0	Timer 0 External Clock Input
P3.5/T1	Timer 1 External Clock Input
P3.6/WR	Write Strobe for External Memory
P3.7/RD	Read Strobe for External Memory
P4.0/SEPCLK	Source for Serial Expansion Port
P4.1/SEPDAT	Data I/O for the Serial Expansion Port
P4.2/C1	PCA1 External Clock Input
P4.3/C1EX0 - P4.7/C1EX4	PCA1 Capture Input, Compare/PWM Output
P5.2/INT2 - P5.6/INT6	External Interrupt INT2 - INT6

2.9 Interrupts

The 80C51GB has 15 interrupt sources, 7 from external interrupt pins (INT0, INT1, INT2, INT3, INT4, INT5, INT6), 3 from timer counters (Timer 0, Timer 1, Timer 2), 2 from programmable counter arrays (PCA and PCA1), and one each from the serial port, A/D converter, and serial expansion port.

INTERRUPT SOURCES

The external interrupts, INT0, and INT1 can be programmed as level-activated (low) or transition-activated (1-to-0). The IT0 and IT1 bits in the TCON Special Function Register control the selection. The interrupt request is actually generated by the IE0 AND IE1 status bits in the TCON register. The IE0 and IE1 bits will be cleared by the on-chip hardware when the service routine is vectored to, only if the interrupt was transition-activated.

INT2 and INT3 can be either positive or negative transition activated. EXICON Special Function Register contains IT2 and IT3 bits to control the selection. The interrupt request is actually generated by the IE2 and IE3 status bits in the EXICON register. The IE2 and IE3 bits will be cleared by the on-chip hardware when the service routine is vectored to.

INT4, INT5 and INT6 are all positive transition activated. The interrupt request is actually generated by the IE4, IE5 and IE6 status bits in the EXICON register. The IE4, IE5 and IE6 bits will be cleared by the on-chip hardware when the service routine is vectored to.

The interrupt of the full-duplex, asynchronous serial port is generated from the logical OR of the RI and TI status bits. The serial port interrupt service routine must read these bits to determine whether the receiver, transmitter, or both are requesting an interrupt.

Similarly, the Timer 2 interrupt is generated from the logical OR of the TF2 and EXF2 bits, and the PCA interrupt is generated from the logical OR of the CF, CCF0, CCF1, CCF2, CCF3, and CCF4 status bits.

The interrupt of the synchronous serial expansion port (SEP) is generated from the SEPIF bit in SEPSTAT register. This bit is set when a read or write operation is completed, or an attempt is made to modify the mode of the read/write transaction when a read/write operation is in progress. See Serial Expansion Port section for further explanation.

The status bits requesting an interrupt must remain set until the interrupt service routine begins. If not done so by hardware, the status bits should be cleared before the termination of the interrupt service routine.

The interrupt sources can be individually enabled by setting their corresponding interrupt enable bits. They can be globally disabled by clearing the EA bit in the IE register.

INTERRUPT PRIORITY

Four priority levels are available for each interrupt source. The priority levels of an interrupt source can be programmed by setting or clearing the corresponding bits in either IP and IP1 or IPA and IPA1 registers as shown below:

Table 17. Interrupt Priority Levels

Bits		Interrupt Priority Level
Pxx.1	Pxx	
0	0	0 (Lowest)
0	1	1
1	0	2
1	1	3 (Highest)

(e.g. PAD=1 and PAD.1=1 will set A/D converter interrupt priority level to 3, while PS=0 and PS.1=1 will set serial port interrupt priority level to 2.)

An interrupt source programmed to the higher level will interrupt service routines of an interrupt source programmed to the lower level. Interrupt sources at the same level will not interrupt each other. When more than one interrupt source on the same level have pending interrupt requests (requests not yet serviced), the priority shown in Table 17 determines which of the pending requests will be serviced first.

INTERRUPT VECTOR ADDRESS AND RETI

When an interrupt is serviced, the processor executes an LCALL instruction to the vector address corresponding to the interrupt source. This begins the interrupt service routine. The last instruction of the service routine should be the RETI instruction. This instruction, in addition to performing a subroutine return (POP stack into PC), enables the interrupt logic to accept additional interrupts.

Table 18. Interrupt Vector Address Handling

Interrupt Source	Interrupt Request Bits	Cleared by Hardware	Vector Address	Priority within Level
INT0	IE0	No (Level) Yes (Trans.)	0003H	1
INT1	IE1	No (Level) Yes (Trans.)	0013H	7
INT2	IE2	YES	0053H	3
INT3	IE3	YES	005BH	6
INT4	IE4	YES	0063H	9
INT5	IE5	YES	006BH	12
INT6	IE6	YES	0073H	15
Timer 0	TF0	YES	000BH	4
Timer 1	TF1	YES	001BH	10
Timer 2	TF2,EXF2	NO	002BH	14
Serial Port	RI,TI	NO	0023H	13
PCA	CF,CCF0,...,CCF4	NO	0043H	11
PCA1	CF1,C1CF0,...,C1CF4	NO	0033H	5
A/D	AIF	NO	003BH	8
SEP	SEPIF	NO	004BH	2

INTERRUPT SPECIAL FUNCTION REGISTERS

Table 19. Bit Assignments

Name	MSB				LSB			
IE	EA	EC	ET2	ES	ET1	EX1	ET0	EX0
IEA	EAD	EX6	EX5	EX4	EX3	EX2	EC1	ESEP
IP	—	PC	PT2	PS	PT1	PX1	PT0	PX0
IP1	—	PC.1	PT2.1	PS.1	PT1.1	PX1.1	PT0.1	PX0.1
IPA	PAD	PX6	PX5	PX4	PX3	PX2	PC1	PSEP
IPA1	PAD.1	PX6.1	PX5.1	PX4.1	PX3.1	PX2.1	PC1.1	PSEP.1
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
CCON	CF	CR	CCF4	CCF3	CCF2	CCF1	CCF0	CCF0
C1CON	CF1	CR1	CRE	C1CF4	C1CF3	C1CF2	C1CF1	C1CF0
ACON	—	—	AIF	ACE	ACS0	ACS1	AIM	ATM
SEPSTAT	—	—	—	—	—	SEPFWR	SEPFRD	SEPIF
EXICON	—	IE6	IE5	IE4	IE3	IE2	IT3	IT2

Table 20. Bit Descriptions

EA	Enable All. This bit is set by software to enable interrupts. When EA is low, all interrupt sources are disabled regardless of their individual interrupt enable bits. When EA is high, interrupt requests from enabled sources will be serviced.
EC	PCA interrupt enable. This bit is set by software to enable PCA interrupt requests.
ET2	Enable Timer 2 interrupts. This bit is set by software to enable interrupt requests from Timer 2.
ES	Enable Serial port interrupts. This bit is set by software to enable interrupt requests from the serial port.
ET1	Enable Timer 1 interrupts. This bit is set by software to enable interrupt requests from Timer 1.
EX1	Enable external interrupt 1. This bit is set by software to enable interrupt requests from INT1.
ET0	Enable Timer 0 interrupts. This bit is set by software to enable interrupt requests from Timer 0.
EX0	Enable external interrupt 0. This bit is set by software to enable interrupt request from INT0.
EAD	Enable A/D converter interrupts. This bit is set by software to enable interrupt requests from A/D converter.
EX6	Enable external interrupt 6. This bit is set by software to enable interrupt requests from INT6.
EX5	Enable external interrupt 5. This bit is set by software to enable interrupt requests from INT5.
EX4	Enable external interrupt 4. This bit is set by software to enable interrupt requests from INT4.
EX3	Enable external interrupt 3. This bit is set by software to enable interrupt requests from INT3.
EX2	Enable external interrupt 2. This bit is set by software to enable interrupt requests from INT2.
EC1	PCA1 interrupt enable. This bit is set by software to enable PCA1 interrupt requests.
ESEP	Enable serial expansion port interrupt. This bit is set by software to enable interrupt requests priority level.
PC	Priority bit 0 for the PCA. This bit and PC.1 are set by software to program the PCA0 interrupt priority level.
PC.1	Priority bit 1 for the PCA. This bit and PC are set by software to program the PCA interrupt priority level.
PT2	Priority bit 0 for Timer 2. This bit and PT2 are set by software to program the Timer 2 interrupt priority level.
PT2.1	Priority bit 1 for Timer 2. This bit and PT2 are set by software to program the Timer 2 interrupt priority level.
PS	Priority bit 0 for the serial port. This bit and PS.1 are set by software to program the serial port interrupt priority level.
PS.1	Priority bit 1 for the serial port. This bit and PS are set by software to program the serial port interrupt priority level.
PT1	Priority bit 0 for Timer 1. This bit and PT1.1 are set by software to program Timer 1 interrupt priority level.

Table 20. Bit Descriptions (Continued)

PT1.1	Priority bit 1 for Timer 1. This bit and PT1 are set by software to program Timer 1 interrupt priority level.
PX1	Priority bit 0 for external interrupt 1. This bit and PX1.1 are set by software to program INT1 interrupt priority level.
PX1.1	Priority bit 1 for external interrupt 1. This bit and PX1 are set by software to program INT1 interrupt priority level.
PT0	Priority bit 0 for Timer 0. This bit and PT0.1 are set by software to program Timer 0 interrupt priority level.
PT0.1	Priority bit 1 for Timer 0. This bit and PT0 are set by software to program Timer 0 interrupt priority level.
PX0	Priority bit 0 for external interrupt 0. This bit and PX0.1 are set by software to program Timer 0 interrupt priority level.
PX0.1	Priority bit 1 for external interrupt 0. This bit and PX0 are set by software to program INT0 interrupt priority level.
PAD	Priority bit 0 for A/D converter. This bit and PAD.1 are set by software to program A/D converter interrupt priority level.
PAD.1	Priority bit 1 for A/D converter. This bit and PAD are set by software to program A/D converter interrupt priority level.
PX6	Priority bit 0 for external interrupt 6. This bit and PX6.1 are set by software to program INT6 interrupt priority level.
PX6.1	Priority bit 1 for external interrupt 6. This bit and PX6 are set by software to program INT6 interrupt priority level.
PX5	Priority bit 0 for external interrupt 5. This bit and PX5.1 are set by software to program INT5 interrupt priority level.
PX5.1	Priority bit 1 for external interrupt 5. This bit and PX5 are set by software to program INT5 interrupt priority level.
PX4	Priority bit 0 for external interrupt 4. This bit and PX4 are set by software to program INT4 interrupt priority level.
PX4.1	Priority bit 1 for external interrupt 4. This bit and PX4 are set by software to program INT4 interrupt priority level.
PX3	Priority bit 0 for external interrupt 3. This bit and PX3.1 are set by software to program INT3 interrupt priority level.
PX3.1	Priority bit 1 for external interrupt 3. This bit and PX3 are set by software to program INT3 interrupt priority level.
PX2	Priority bit 0 for external interrupt 2. This bit and PX2.1 are set by software to program INT2 interrupt priority level.
PX2.1	Priority bit 1 for external interrupt 2. This bit and PX2 are set by software to program INT2 interrupt priority level.
PC1	Priority bit 0 for the PCA1. This bit and PC1 are set by software to program the PCA1 interrupt priority level.
PC1.1	Priority bit 1 for the PCA1. This bit and PC1 are set by software to program the PCA1 interrupt priority level.
PSEP	Priority bit 0 for serial expansion port. This bit and PSEP.1 are set by software to program serial expansion port interrupt priority level.

Table 20. Bit Descriptions (Continued)

PSEP.1	Priority bit 0 for serial expansion port. This bit and PSEP are set by software to program serial expansion port interrupt priority level.
TF1	Timer 1 overflow Flag. This bit is set by hardware when Timer 1 overflows. It is cleared by hardware when the processor vectors to the Timer 1 interrupt routine.
TR1	See section 2.4 on the Timer/Counters
TF0	Timer 0 overflow Flag. This bit is set by hardware when Timer 0 overflows. It is cleared by hardware when the processor vectors to the Timer 0 interrupt routine.
TR0	See section 2.4 on the Timer/Counters
IE1	Interrupt 1 Edge flag. This bit is set by hardware when an external interrupt edge is detected. It is cleared by hardware when the processor vectors to the INT1 interrupt routine.
IT1	Interrupt 1 Type control bit. This bit is set or cleared by software to control whether INT1 is level or edge triggered. When IT1 is high, an interrupt is triggered by a falling edge on INT1. When IT1 is low, an interrupt is triggered by a low level on INT1.
IE0	Interrupt 0 Edge flag. This bit is set by hardware when an external interrupt edge is detected. It is cleared by hardware when the processor vectors to the INT0 interrupt routine.
IT0	Interrupt 0 Type control bit. This bit is set or cleared by software to control whether INT0 is level or edge triggered. When IT0 is high, an interrupt is triggered by a falling edge on INT0. When IT0 is low, an interrupt is triggered by a low level on INT0.
TF2	Timer 2 overflow flag. This bit is set by hardware when Timer 2 overflows. It can initiate an interrupt request if enabled. TF2 is not cleared by hardware, so it should be cleared in the Timer 2 interrupt routine.
EXF2	Timer 2 External Flag. This bit is set by hardware when a capture or reload is caused by the T2EX pin. It can initiate an interrupt request if enabled. EXF2 is not cleared by hardware, so it should be cleared in the Timer 2 interrupt routine.
RCLK	See Timer/Counters section.
TCLK	See Timer/Counters section.
EXEN2	See Timer/Counters section.
TR2	See Timer/Counters section.
C/T2	See Timer/Counters section.
CP/RL2	See Timer/Counters section.
CF	Counter Flag. This bit is set high by hardware when the PCA Counter overflows.
CR	See PCA section.
CCF4-0	Capture Compare Flags. These five bits correspond to the five Register/Comparator Modules. When a capture or comparison match occurs in one of the Register/Comparator Modules the corresponding CCFn bit is set high.
CF1	Counter Flag. This bit is set high by hardware when the PCA1 Counter overflows.
CR1	See PCA1 section.
CRE	See PCA1 section.
C1CF4-10	Capture Compare Flags. These five bits correspond to the five Register/Comparator Modules. When a capture or comparison match occurs in one of the Register/Comparator Modules, the corresponding CCF1n bit is set high.
AIF	A/D Interrupt Flag. This bit is set by the A/D hardware when an A/D converter interrupt is generated.
ACE	See A/D Converter section.

ACS0	See A/D Converter section.
ACS1	See A/D Converter section.
AIM	See A/D Converter section.
ATM	See A/D Converter section.
SEPFIF	SEP Interrupt Flag. This bit is set by the SEP hardware when an SEP interrupt is generated.
SEPFWR	See serial expansion port section.
SEPFRD	See serial expansion port section.
IE6	Interrupt 6 Edge flag. This bit is set by hardware when an external interrupt edge is detected. It is cleared by hardware when the processor vectors to the INT6 interrupt routine.
IE5	Interrupt 5 Edge flag. This bit is set by hardware when an external interrupt edge is detected. It is cleared by hardware when the processor vectors to the INT5 interrupt routine.
IE4	Interrupt 4 Edge flag. This bit is set by hardware when an external interrupt edge is detected. It is cleared by hardware when the processor vectors to the INT4 interrupt routine.
IE3	Interrupt 3 Edge flag. This bit is set by hardware when an external interrupt edge is detected. It is cleared by hardware when the processor vectors to the INT3 interrupt routine.
IE2	Interrupt 2 Edge flag. This bit is set by hardware when an external interrupt edge is detected. It is cleared by hardware when the processor vectors to the INT2 interrupt routine.
IT3	Interrupt 3 Type control bit. This bit is set or cleared by software to control whether INT3 is positive or negative transition activated. When IT3 is high, an interrupt is triggered by a positive-transition (0-1) on INT3. When IT3 is low, an interrupt is triggered by a negative-transition (1-0) on INT3.
IT2	Interrupt 2 Type control bit. This bit is set or cleared by software to control whether INT2 is positive or negative transition activated. When IT2 is high, an interrupt is triggered by a positive-transition (0-1) on INT2. When IT2 is low, an interrupt is triggered by a negative-transition (1-0) on INT2.

2.10 Power Reduction Modes

The 80C51GB has two power reduction modes, Idle and Power Down. In the Idle mode, the CPU functions are suspended, while the peripheral functions continue to operate normally. In the Power Down mode, both CPU and peripheral functions are suspended, affording the greatest power savings. The user must disable the Oscillator Fail Detect before entering Power Down Mode. If the Oscillator Fail Detect is not disabled before entering Power Down Mode the Oscillator Fail will reset the part and this will bring the 80C51GB out of Power Down.

IDLE

In the Idle mode, the clock signals to the CPU are stopped, thereby suspending the CPU operation. The

CPU status is preserved during the Idle mode. The RAM and all registers are also preserved. The interrupts, serial port, timers, and PCA (a programmable option) continue to operate normally. The I/O port pins hold the logical states they had at the time Idle was activated. ALE and PSEN hold at high logic levels.

The Idle mode is initiated by setting the IDL bit in the PCON Special Function Register. Program execution ends with the completion of this instruction. The Idle mode is terminated by the occurrence of an enabled interrupt on the INT0 or INT1 pins only or by reset. When the Idle mode is terminated by an interrupt, the interrupt service routine is executed. Upon completion of the service routine, execution continues with the instruction following the one which initiated the Idle mode. When Idle is terminated by reset, The Special Function Registers are initialized and execution begins at address 0000H.

POWER DOWN

In the Power Down mode, the CPU and peripheral functions are suspended, and the crystal oscillator is disabled. As in the Idle mode, the CPU state is preserved along with the registers, I/O pins, and RAM. ALE and PSEN are held at low logic levels.

Power Down is initiated by setting the PD bit in the PCON register. Like the Idle mode, program execution ends with the completion of this instruction. The Power Down mode is terminated by the occurrence of an enabled interrupt on the INT0 or INT1 pins only or by reset.

Upon termination of the Power Down mode, the crystal oscillator is enabled by the leading edge of the interrupt or reset signal. The CPU and peripherals are enabled by the trailing edge. The interrupt or reset pulse width must be at least 10 ms to allow the oscillator to restart and stabilize.

GF0 AND GF1

The GF0 and GF1 bits of the PCON register can be used to give an indication if an interrupt occurred during normal execution, or during the Idle or Power Down mode. For example, an instruction which activates Idle can also set one or both GF bits. When Idle is terminated by an interrupt, the interrupt service routine can examine both GF bits.

POF

The POF bit in the PCON register can be used by a reset initialization routine to determine if the 80C51GB has just been powered up. The POF bit is set by the rising edge of V_{CC} and can be cleared by software when the RAM and registers have been initialized.

POWER CONTROL SPECIAL FUNCTION REGISTERS

Table 21. Bit Assignments

Name	Bit							
	MSB							LSB
	7	6	5	4	3	2	1	0
PCON	SMOD1	SMOD2	—	POF	GF1	GF0	PD	IDL

2.11 80C51GB External Memory Addressing

In addition to internal memory accesses, the 80C51GB allows for external program and data memory accessing. The implementation follows that of the 80C51. The program and data memory sections are each 64K bytes long. The first 8K of the program address space is used by on-chip EPROM. When EA (External Address enable) is tied low and the security bits are not programmed, the first 8K of the program address space will be mapped external (no internal EPROM accesses). The MOVX instruction is used for external data reads and writes, and the MOVC can be used to read program code.

For data transfers, R0 or R1 can be used as a pointer to the first page of external data memory. Read and write instructions are coded as follows:

```
MOVX A,@Ri (i=0,1) ; A ← ((Ri))
MOVX @Ri,A ; ((Ri)) ← A
```

The MOVX A,@Ri instructions allow 256 bytes of external data memory to be addressed with just Port 0, ALE (Address Latch enable), RD and WR. Port 2 is not affected. Port 0 is multiplexed with address and data.

Table 22. Bit Descriptions

SMOD1	See Serial Port section
SMOD0	See Serial Port section
POF	Power-on Flag bit. This bit is set by the rising edge of V_{CC} and cleared by software.
GF1-0	General purpose Flag bits. These bits are general purpose bits which can be set or cleared by software.
PD	Power Down bit. This bit is set by software to initiate the Power Down mode. It is cleared by hardware when Power Down is terminated by an interrupt or reset.
IDL	Idle bit. This bit is set by software to initiate the Idle mode. It is cleared by hardware when Idle is terminated by an interrupt or reset.

The DPTR 16-bit register can be used to access any byte in the 64K external data address space. Use of port pins and paging is not necessary to address >256 bytes. The MOVX A,@DPTR instructions use both Port 0 and Port 2 for address. The instructions follow the same pattern as previously shown.

In addition to opcode fetches, the program code memory can be accessed with the MOVC instruction. The MOVC instruction uses A as an index to either DPTR or PC. Only read cycles are allowed and PSEN (Program Store enable) is used to strobe a program memory read. For a common program and data memory, PSEN and RD can be negative ORed for a single read strobe. The instruction format for program memory accessed are shown below:

```
MOVC A,@+PC      ; A ← ((A + PC))
MOVC A,@A+DPTR   ; A ← ((A + DPTR))
```

There are reasons for using MOVC A,@DPTR and MOVC A,@A+DPTR for external tables. MOVC is the quickest since the exact table location is calculated in hardware. For the fastest table access, the MOVC A,@A+DPTR,PC instructions are preferred. The access time required by the data memory instructions is much relaxed as compared with EPROM access time. Therefore in low cost lookup applications, slow EPROMs addressed by MOVX instructions are probably more cost effective.

2.12 Special Function Registers

Most on-chip resources of the 80C51GB are controlled through the Special Function Registers. These registers are accessed by any of the "direct" instructions. For instance, the instruction

```
MOV #A,direct
```

where "direct" is the Port 1 register address, will input a byte from Port 1 and load it into the Accumulator. The Special Function Register addresses are in the range 128 to 255.

Some of the Special Function Registers are bit addressable as well as byte addressable. Individual bits in these registers can be accessed, i.e. moved to or from the carry bit or tested using the "direct bit" instructions. For instance, the instruction

```
SETB bit
```

where "bit" is the bit address of port pin P2.3, will write a logic 1 to port pin P2.3.

The Special Function Register address assignments and value initialized by reset are listed below.

Table 23. Special Function Register Addresses

Name	Description	SFR Address	Initial Value	Bit Addressable
ACC	Accumulator	E0H	00000000	Yes
ACMP	A/D Compare Register	C6H	00000000	No
ACON	A/D Control	97H	00000000	No
AD0	A/D Result	84H	00000000	No
AD1	A/D Result	94H	00000000	No
AD2	A/D Result	A4H	00000000	No
AD3	A/D Result	B4H	00000000	No
AD4	A/D Result	C4H	00000000	No
AD5	A/D Result	D4H	00000000	No
AD6	A/D Result	E4H	00000000	No
AD7	A/D Result	F4H	00000000	No
AUXR	Auxiliary Register	8EH	XXXXXXX0	No
B	B Register	F0H	00000000	Yes
CCON	PCA Control	D8H	00X00000	Yes
C1CON	PCA 1 Control	E8H	00X00000	Yes
CMOD	PCA Mode	D9H	00XXX000	No

Table 23. (Continued)

Name	Description	SFR Address	Initial Value	Bit Addressable
C1MOD	PCA 1 Mode	DFH	00XX0000	No
CCAPM0	PCA Mode	DAH	X0000000	No
CCAPM1	PCA Mode	DBH	X0000000	No
CCAPM2	PCA Mode	DCH	X0000000	No
CCAPM3	PCA Mode	DDH	X0000000	No
CCAPM4	PCA Mode	DEH	X0000000	No
C1CAPM0	PCA 1 Mode	9AH	X0000000	No
C1CAPM1	PCA 1 Mode	9BH	X0000000	No
C1CAPM2	PCA 1 Mode	9CH	X0000000	No
C1CAPM3	PCA 1 Mode	9DH	X0000000	No
C1CAPM4	PCA 1 Mode	9EH	X0000000	No
CCAP0H	PCA Compare/Capture High	FAH	XXXXXXXX	No
CCAP1H	PCA Compare/Capture High	FBH	XXXXXXXX	No
CCAP2H	PCA Compare/Capture High	FCH	XXXXXXXX	No
CCAP3H	PCA Compare/Capture High	FDH	XXXXXXXX	No
CCAP4H	PCA Compare/Capture High	FEH	XXXXXXXX	No
C1CAP0H	PCA 1 Compare/Capture High	BAH	XXXXXXXX	No
C1CAP1H	PCA 1 Compare/Capture High	BBH	XXXXXXXX	No
C1CAP2H	PCA 1 Compare/Capture High	BCH	XXXXXXXX	No
C1CAP3H	PCA 1 Compare/Capture High	BDH	XXXXXXXX	No
C1CAP4H	PCA 1 Compare/Capture High	BEH	XXXXXXXX	No
CCAP0L	PCA Compare/Capture Low	EAH	XXXXXXXX	No
CCAP1L	PCA Compare/Capture Low	EBH	XXXXXXXX	No
CCAP2L	PCA Compare/Capture Low	ECH	XXXXXXXX	No
CCAP3L	PCA Compare/Capture Low	EDH	XXXXXXXX	No
CCAP4L	PCA Compare/Capture Low	EEH	XXXXXXXX	No
C1CAP0L	PCA 1 Compare/Capture Low	AAH	XXXXXXXX	No
C1CAP1L	PCA 1 Compare/Capture Low	ABH	XXXXXXXX	No
C1CAP2L	PCA 1 Compare/Capture Low	ACH	XXXXXXXX	No
C1CAP3L	PCA 1 Compare/Capture Low	ADH	XXXXXXXX	No
C1CAP4L	PCA 1 Compare/Capture Low	AEH	XXXXXXXX	No
CH	PCA Counter High	F9H	00000000	No
CH1	PCA 1 Counter High	EFH	00000000	No
CL	PCA Counter Low	E9H	00000000	No
CL1	PCA 1 Counter Low	FFH	00000000	No
DPH	DPTR High Byte	83H	00000000	No
DPL	DPTR Low Byte	82H	00000000	No
EXICON	External Interrupt Control	C7H	00000000	No

Table 23. (Continued)

Name	Description	SFR Address	Initial Value	Bit Addressable
IE	Interrupt Enable	A8H	00000000	Yes
IEA	Interrupt Enable A	A7H	00000000	No
IP	Interrupt Priority	B8H	X0000000	Yes
IPA	Interrupt Priority A	B6H	00000000	No
IPA1	Interrupt Priority A1	B5H	00000000	No
OSCR	Oscillator Fail Detect	A5H	XXXXXXX0	No
PCON	Power Control	87H	00XX0000	No
PSW	Program Status Word	D0H	00000000	Yes
P0	Port 0	80H	11111111	Yes
P1	Port 1	90H	11111111	Yes
P2	Port 2	A0H	00000000	Yes
P3	Port 3	B0H	11111111	Yes
P4	Port 4	C0H	00000000	Yes
P5	Port 5	F8H	00000000	Yes
RCAP2H	Timer 2 Reload/Capture High	CBH	00000000	No
RCAP2L	Timer 2 Reload/Capture Low	CAH	00000000	No
SADDR	Serial Port Slave Address	A9H	00000000	No
SADEN	Serial Port Address Enable	B9H	00000000	No
SBUF	Serial Port / Transmit	99H	XXXXXXXX	No
SCON	Serial Port Control	98H	00000000	Yes
SEPCON	Serial Expansion Port Control	D7H	00000000	No
SEPSTAT	Serial Expansion Port STATUS	F7H	XXXXXXXX	No
SP	Stack Pointer	81H	00000111	No
TCON	Timer 0,1 Control	88H	00000000	Yes
TMOD	Timer 0,1 Mode	89H	00000000	No
T2CON	Timer 2 Control	C8H	00000000	Yes
T2MOD	Timer 2 Mode	C9H	XXXXXXX0	No
T0H	Timer 0 High	8CH	00000000	No
T0L	Timer 0 Low	8AH	00000000	No
T1H	Timer 1 High	8DH	00000000	No
T1L	Timer 1 Low	8BH	00000000	No
T2H	Timer 2 High	CDH	00000000	No
T2L	Timer 2 Low	CCH	00000000	No
WDT	Watchdog Timer Reset	A6H	XXXXXXXX	No

Table 24. SFR Mapping and Initial Contents

L	H						
	8	9	A	B	C	D	E F
0	P0 * 11111111	P1 * 11111111	P2 * 00000000	P3 * 11111111	P4 * 00000000	PSW * 00000000	A * 00000000 B * 00000000
1	SP 00000111						
2	DPL 00000000						
3	DPH 00000000						
4	AD0 00000000	AD1 00000000	AD2 00000000	AD3 00000000	AD4 00000000	AD5 00000000	AD6 00000000 AD7 00000000
5			OFDDIS XXXXXXXX0	IPA1 00000000			
6			WDTRST XXXXXXXX	IPA 00000000	EXICON 00000000		
7	PCON 00XX0000	ACON XX000000	IEA 00000000	IP1 00000000	ACMP 00000000	SEPCON 000000	SEPD XXXXXXXXX SEPSTAT 000
8	TCON * 00000000	SCON * 00000000	IE * 00000000	IP * X0000000	T2CON * 00000000	CCON * 00X00000	CCON1 * 00X00000 P5 * 00000000
9	TMOD 00000000	SBUF XXXXXXXX	SADDR 00000000	SADEN 00000000	T2MOD XXXXXXXX0	CMOD 00XX0000	CL 00000000 CH 00000000
A	TL0 00000000	C1CAPM0 X0000000	C1CAP0L XXXXXXXX	C1CAP0H XXXXXXXX	RCAP2L 00000000	CCAPM0 X0000000	CCAPOL XXXXXXXX CCAP0H XXXXXXXX
B	TL1 00000000	C1CAPM1 X0000000	C1CAP1L XXXXXXXX	C1CAP1H XXXXXXXX	RCAP2H 00000000	CCAPM1 X0000000	CCAP1L XXXXXXXX CCAP1H XXXXXXXX
C	TH0 00000000	C1CAPM2 X0000000	C1CAP2L XXXXXXXX	C1CAP2H XXXXXXXX	T2L 00000000	CCAPM2 X0000000	CCAP2L XXXXXXXX CCAP2H XXXXXXXX
D	TH1 00000000	C1CAPM3 X0000000	C1CAP3L XXXXXXXX	C1CAP3H XXXXXXXX	T2H 00000000	CCAPM3 X0000000	CCAP3L XXXXXXXX CCAP3H XXXXXXXX
E	AUXR XXXXXXXX0	C1CAPM4 X0000000	C1CAP4L XXXXXXXX	C1CAP4H XXXXXXXX		CCAPM4 X0000000	CCAP4L XXXXXXXX CCAP4H XXXXXXXX
F		C1MOD 00XX0000	CL1 00000000	CH1 00000000			

NOTE:

All blank addresses give undefined results when read or written.

* = Bit addressable register.

ONCE™ MODE

This mode is activated when PSEN is held high and ALE is held low during reset. When RST falls, PSEN and ALE are latched and the 80C51GB remains in this mode until the next reset.

When the part is put into this mode, it becomes inactive. All ports, PSEN, and ALE are pulled high with weak pullups only. But XTAL2 output is not disabled. This mode allows an ICETM bondout chip to be clipped over the part and drive all signals while the part is inactive. The bondout chip drives ALE low during re-

set to put the 80C51GB part in this mode. It is to be used for on board testing.

3.0 PIN DESCRIPTIONS

3.1 Pin Assignment for 68 Lead PLCC Package

The 80C51GB will be packaged in the 68 lead PLCC and CERQUAD package. Its pin assignment is shown below.

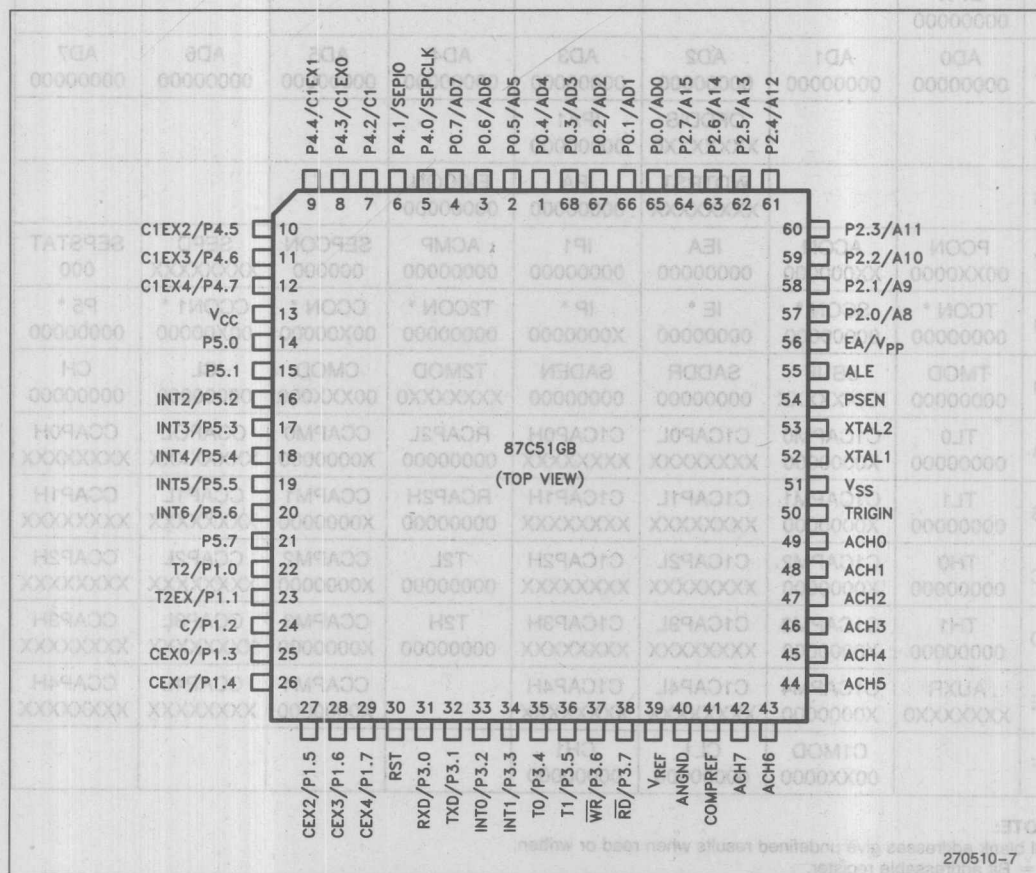


Figure 7. Pin Assignment

4.0 ELECTRICAL CHARACTERISTICS

4.1 DC Characteristics

$T_A = -40^\circ\text{C}$ to $+125^\circ\text{C}$ case temperature, $V_{CC} = 4.5\text{V}$ to 5.5V

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
V_{T+}	High-Going Threshold (except XTAL1, RST)	1.10	1.50	2.0	V	
V_{T-}	Low-Going Threshold (except XTAL1, RST)	0.60	0.90	1.2	V	
V_{HYS}	Hysteresis ($V_{T+} - V_{T-}$) (except XTAL1, RST)	0.40	0.60		V	
V_{IL}	Input Low Voltage (XTAL1, RST)	-0.5		$0.2V_{CC} - 0.1$	V	
V_{IH}	Input High Voltage (XTAL1, RST)	$0.7V_{CC}$		$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage (Ports 1,2,3,4,5)			0.45	V	$I_{OL} = 1.6\text{ mA}$
V_{OL1}	Output Low Voltage (Port 0, ALE, PSEN)			0.45	V	$I_{OL} = 3.2\text{ mA}$
V_{OH}	Output High Voltage (Ports 1,2,3,4,5)	2.4 $0.9V_{CC}$			V V	$I_{OH} = -60\text{ }\mu\text{A}$ $I_{OH} = -10\text{ }\mu\text{A}$
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode, ALE, PSEN)	2.4 $0.9V_{CC}$			V V	$I_{OH} = -800\text{ }\mu\text{A}$ $I_{OH} = -80\text{ }\mu\text{A}$
I_{IL}	Logical 0 Input Current (Ports 1,2,3,4,5)			-50	μA	$V_{IN} = 0.45\text{V}$
I_{TL}	Logical 1-to-0 Transition Current (Ports 1,2,3,4,5)			-650	μA	$V_{IN} = 2.0\text{V}$
I_{LI}	Input Leakage Current (Port 0, EA)			± 10	μA	$0.45 < V_{IN} < V_{CC}$
RRST	RST Pulldown Resistor	40		125	k Ω	
C_{IO}	Pin Capacitance			10	pF	freq = 1 MHz $T_A = 25^\circ\text{C}$
I_{PD}	Power Down Current			200	μA	(2)
I_{DL}	Idle Mode Current			10	μA	(3)
I_{CC}	Operating Current			40	μA	(4)

NOTES:

- Typical values are obtained using $V_{CC} = 5.0\text{V}$ and $T_A = 25^\circ\text{C}$.
- Power Down Current is measured with all output pins disconnected, EA = PORT0 = V_{CC} , XTAL2 N.C., RST = V_{SS} .
- Idle ICC is measured with all output pins disconnected, XTAL1 driven with TCHLCH, TCHLCH = 10 ns, $V_{IL} = V_{SS} + 0.5\text{V}$, $V_{IH} = V_{CC} - 0.5\text{V}$, XTAL2 N.C., EA = PORT0 = V_{CC} , RST = V_{SS} , internal clock to PCA gated off.
- ICC is measured with all output pins disconnected, XTAL1 driven with TCLCH, TCLCH = 10 ns, $V_{IL} = V_{SS} + 0.5\text{V}$, $V_{IH} = V_{CC} - 0.5\text{V}$, XTAL2 N.C., EA = RST = PORT0 = V_{CC} .

4.2 AC Specifications

$T_A = -40^\circ\text{C}$ to $+125^\circ\text{C}$ case temperature, $V_{CC} = 4.5\text{V}$ to 5.5V , Load Capacitance on Port 0, ALE, and $PSEN = 100\text{ pF}$, Load Capacitance on all other outputs = 80 pF

EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12	MHz
TLHLL	ALE Pulse Width	112		2TCLCL - 55		ns
TAVLL	ADDR Valid to ALE Low	13		TCLCL - 70		ns
TALLAX	ADDR Hold after ALE Low	33		TCLCL - 50		ns
TLLIV	ALE Low to Valid Instruction In		218		4TCLCL - 115	ns
TLLPL	ALE Low to $PSEN$ Low	28		TCLCL - 55		ns
TPLPH	$PSEN$ Pulse Width	190		3TCLCL - 60		ns
TPLIV	$PSEN$ Low to Valid Instruction In		130		3TCLCL - 120	ns
TPXIX	Input Instruction Hold after $PSEN$	0		0		ns
TPXIZ	Input Instruction Float after $PSEN$		43		TCLCL - 40	ns
TAVIV	ADDR to Valid Instruction In		297		5TCLCL - 120	ns
TPLAZ	$PSEN$ Low to ADDR Float		25		25	ns
TRLRH	RD Pulse Width	400		6TCLCL - 100		ns
TWLWH	WR Pulse Width	400		6TCLCL - 100		ns
TRLDV	RD Low to Valid Data In		232		5TCLCL - 185	ns
TRHDX	Data Hold after RD	0		0		ns
TRHDZ	Data Float after RD		82		2TCLCL - 85	ns
TLLDV	ALE Low to Valid Data In		496		8TCLCL - 170	ns
TAVDV	ADDR to Valid Data In		565		9TCLCL - 185	ns
TLLWL	ALE Low to RD or WR Low	185	315	3TCLCL - 65	3TCLCL + 65	ns
TAVWL	ADDR to RD or WR Low	188		4TCLCL - 145		ns
TQVWX	Data Valid to WR Transition	8		TCLCL - 75		ns
TWHGX	Data Hold after WR	18		TCLCL - 65		ns
TRLAZ	RD Low to ADDR Float		0		0	ns
TWHLH	RD or WR High to ALE High	18	148	TCLCL - 65	TCLCL + 65	ns

SERIAL PORT TIMING - SHIFT REGISTER MODE

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port CLK CY	1.0		12TCLCL		μ s
TQVXH	Output Data Setup to CLK Rising Edge	700		10TCLCL - 133		ns
TXHGX	Output Data Hold after CLK Rising Edge	50		2TCLCL - 117		ns
TXHDX	Input Data Hold after CLK Rising Edge	0		0		ns
TXHDV	CLK Rising Edge to Input Data Valid		700		10TCLCL - 133	ns

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

4.3 A/D Converter Specifications

$T_A = -40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ case temperature, $V_{CC} = 4.5$ TO 5.5 V, $V_{REF} = 4.5$ TO 5.5 V, $V_{SS} = \text{ANGND} = 0$ V

The absolute conversion accuracy is dependent on the accuracy of the V_{REF} . The specifications given below assume adherence to the operating conditions section of this data sheet.

Testing is done at $V_{REF} = 5.120$ volts

Resolution 8 bits

Accuracy $\pm \frac{1}{2}$ LSB

Differential nonlinearity $\pm \frac{1}{2}$ LSB

Integral nonlinearity ± 1 LSB

Channel to channel matching ± 1 LSB

Cross talk (DC to 100 KHz) -60 dB

**MCS⁵¹ Data Sheets,
Application Notes,
Article Reprint and
Development Support Tools**

14

MCS®-51 Data Sheets,
Application Notes,
Article Reprint and
Development Support Tools

14



PRELIMINARY

MCS[®]-51 8-BIT CONTROL-ORIENTED MICROCOMPUTERS

8031/8051

8031AH/8051AH

8032AH/8052AH

8751H/8751H-8

AUTOMOTIVE

- High Performance HMOS Process
- Internal Timers/Event Counters
- 2-Level Interrupt Priority Structure
- 32 I/O Lines (Four 8-Bit Ports)
- 64K Program Memory Space
- Security Feature Protects EPROM Parts Against Software Piracy
- Boolean Processor
- Bit-Addressable RAM
- Programmable Full Duplex Serial Channel
- 111 Instructions (64 Single-Cycle)
- 64K Data Memory Space
- Available in LCC, PLCC, and DIP Packages

(See Packaging Outline and Dimensions, Order # 231369)

The MCS[®]-51 products are optimized for control applications. Byte-processing and numerical operations on small data structures are facilitated by a variety of fast addressing modes for accessing the internal RAM. The instruction set provides a convenient menu of 8-bit arithmetic instructions, including multiply and divide instructions. Extensive on-chip support is provided for one-bit variables as a separate data type, allowing direct bit manipulation and testing in control and logic systems that require Boolean processing.

Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16-Bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8032AH	none	256 x 8 RAM	3 x 16-Bit	6
8031AH	none	128 x 8 RAM	2 x 16-Bit	5
8031	none	128 x 8 RAM	2 x 16-Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5
8751H-8	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5

The 8751H is an EPROM version of the 8051AH; that is, the on-chip Program Memory can be electrically programmed, and can be erased by exposure to ultraviolet light. It is fully compatible with its predecessor, the 8751-8, but incorporates two new features: a Program Memory Security bit that can be used to protect the EPROM against unauthorized read-out, and a programmable baud rate modification bit (SMOD). The 8751H-8 is identical to the 8751H but only operates up to 8 MHz.

PRODUCT OPTIONS

Intel's extended and automotive temperature range products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to +70°C ambient. With the extended temperature range option, operational characteristics are guaranteed over the temperature

range of -40°C to +85°C ambient. For the automotive temperature range option, operational characteristics are guaranteed over the temperature range of -40°C to +110°C ambient.

The automotive, extended, and commercial temperature versions of the MCS-51 product families are available with or without burn-in options as listed in Table 1.

As shown in Figure 1, temperature, burn-in, and package options are identified by a one- or two-letter prefix to the part number.

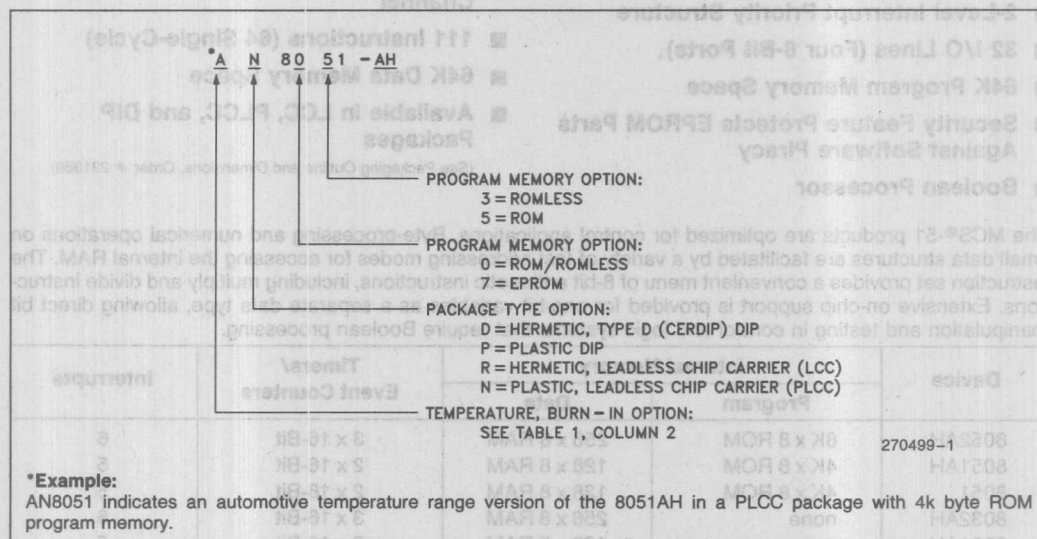


Figure 1. MCS®-51 Product Family Nomenclature

Table 1. Temperature Burn-In Options

Temperature Classification	Temperature Designation	Operating Temperature °C Ambient	Burn-In 125°C (Hr)
Commercial	Null Q	0 to +70 0 to +70	None 168 ± 8
Extended	T L	-40 to +85 -40 to +85	None 168 ± 8
Automotive	A	-40 to +110	None

NOTE:

- Other burn-in options are also available, but not standard.

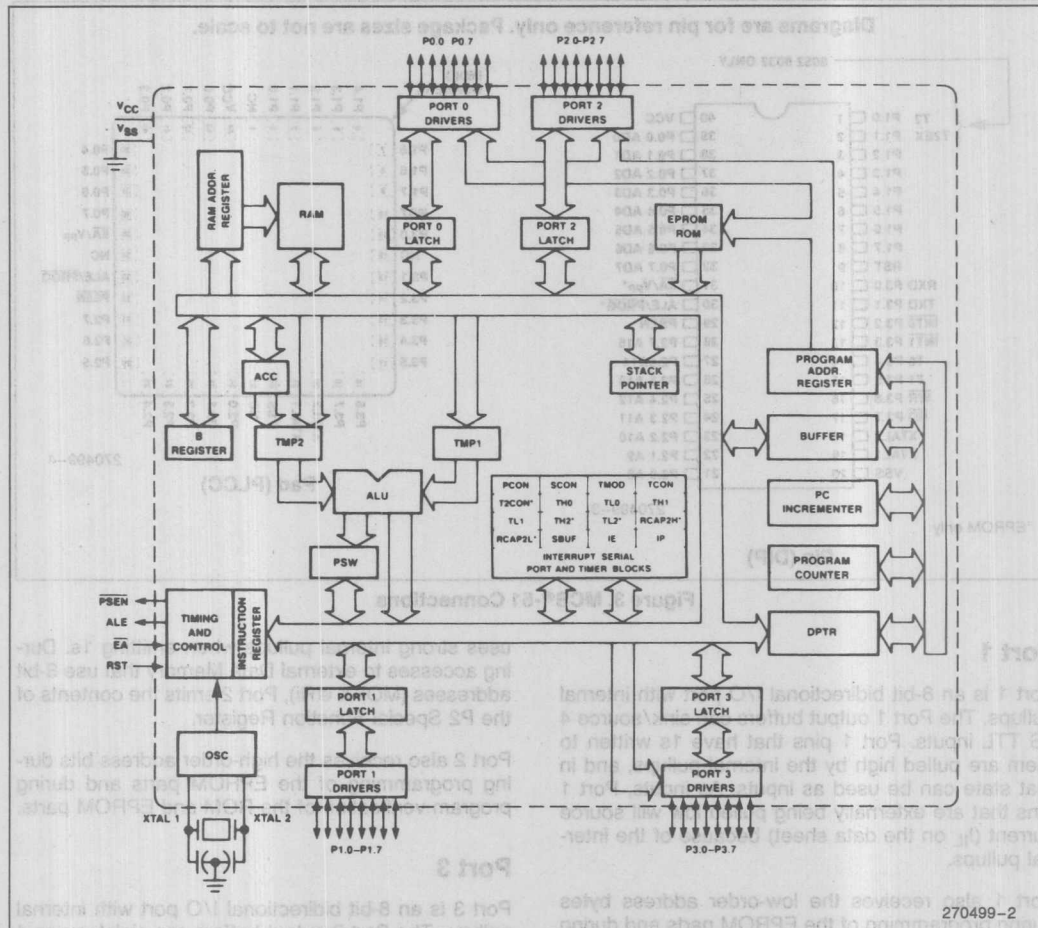


Figure 2. MCS®-51 Block Diagram

PIN DESCRIPTIONS

VCC

Supply voltage.

VSS

Circuit ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS² TTL inputs.

Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during programming of the EPROM parts, and outputs the code bytes during program verification of the ROM and EPROM parts. External pullups are required during program verification.

Diagrams are for pin reference only. Package sizes are not to scale.

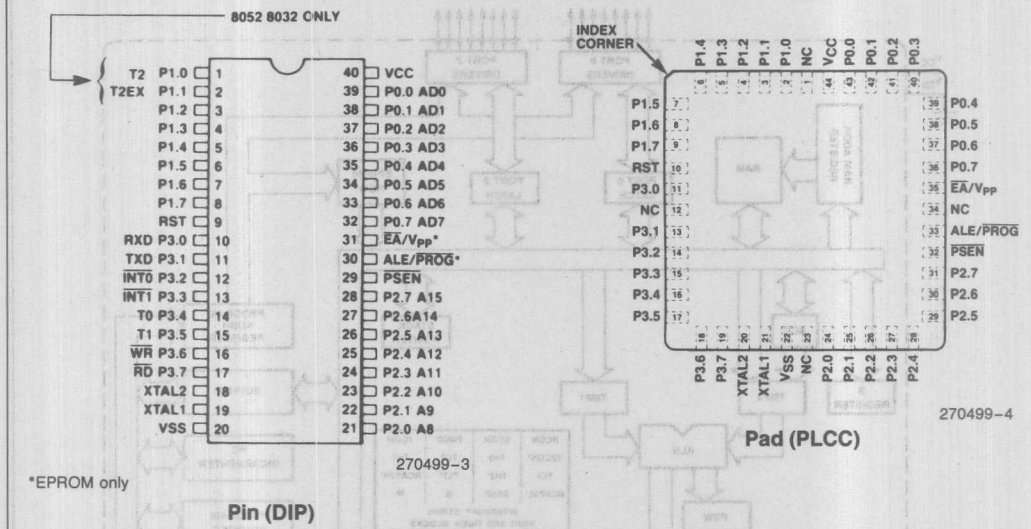


Figure 3. MCS[®]-51 Connections

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

In the 8032AH and 8052AH, Port 1 pins P1.0 and P1.1 also serve the T2 and T2EX functions, respectively.

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it

uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/PROG

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during programming of the EPROM parts.

In normal operation ALE is emitted at a constant rate of $\frac{1}{6}$ the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

PSEN

Program Store Enable is the read strobe to external Program Memory.

When the device is executing code from external Program Memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external Data Memory.

\overline{EA}/V_{PP}

External Access enable \overline{EA} must be strapped to V_{SS} in order to enable any MCS-51 device to fetch code from external Program memory locations 0 to 0FFFF (0 to 1FFFF, in the 8032AH and 8052AH).

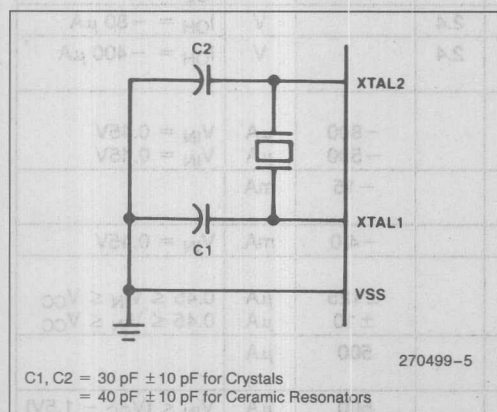


Figure 4. Oscillator Connections

Note, however, that if the Security Bit in the EPROM devices is programmed, the device will not fetch code from any location in external Program Memory.

This pin also receives the 21V programming supply voltage (V_{PP}) during programming of the EPROM parts.

XTAL1

Input to the inverting oscillator amplifier.

XTAL2

Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 4. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 5. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

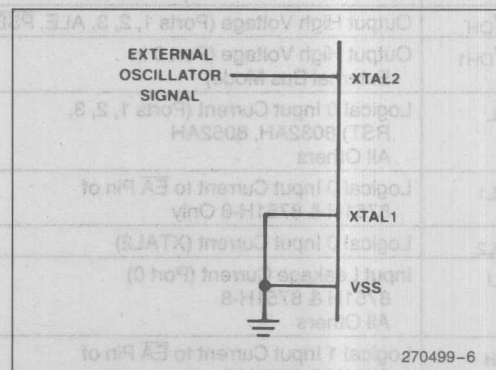


Figure 5. External Drive Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature	Under Bias	−40°C to +110°C
Storage Temperature		−65°C to +150°C
Voltage on $\bar{E}A/VPP$ Pin to V_{SS}		−0.5V to +21.5V
Voltage on Any Other Pin to V_{SS}		−0.5V to +7V
Power Dissipation		1.5W
Maximum Case Temperature	Under bias	+125°C

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Typical Thermal Resistance Junction to Ambient (θ_{ja})

Package	θ_{ja}
Plastic	75°C/W
CERDIP	36°C/W
PLCC	46°C/W

D.C. CHARACTERISTICS $T_A = -40^\circ\text{C}$ to $+110^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (Except $\bar{E}A$ Pin of 8751H & 8751H-8)	−0.5	0.7	V	
V_{IL1}	Input Low Voltage to $\bar{E}A$ Pin of 8751H & 8751H-8	0	0.6	V	
V_{IH}	Input High Voltage (Except XTAL2, RST)	2.0	$V_{CC} + 0.5$	V	
V_{IH1}	Input High Voltage to XTAL2, RST	2.5	$V_{CC} + 0.5$	V	XTAL1 = V_{SS}
V_{OL}	Output Low Voltage (Ports 1, 2, 3)*		0.45	V	$I_{OL} = 1.6 \text{ mA}$
V_{OL1}	Output Low Voltage (Port 0, ALE, PSEN)*				
	8751H, 8751H-8		0.60	V	$I_{OL} = 3.2 \text{ mA}$
			0.45	V	$I_{OL} = 2.4 \text{ mA}$
	All Others		0.45	V	$I_{OL} = 3.2 \text{ mA}$
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, PSEN)	2.4		V	$I_{OH} = -80 \mu\text{A}$
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	$I_{OH} = -400 \mu\text{A}$
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, RST) 8032AH, 8052AH		−800	μA	$V_{IN} = 0.45V$
	All Others		−500	μA	$V_{IN} = 0.45V$
I_{IL1}	Logical 0 Input Current to $\bar{E}A$ Pin of 8751H & 8751H-8 Only		−15	mA	
I_{IL2}	Logical 0 Input Current (XTAL2)		−4.0	mA	$V_{IN} = 0.45V$
I_{LI}	Input Leakage Current (Port 0)				
	8751H & 8751H-8		± 125	μA	$0.45 \leq V_{IN} \leq V_{CC}$
	All Others		± 10	μA	$0.45 \leq V_{IN} \leq V_{CC}$
I_{IH}	Logical 1 Input Current to $\bar{E}A$ Pin of 8751H & 8751H-8		500	μA	
I_{IH1}	Input Current to RST to Activate Reset		500	μA	$V_{IN} < (V_{CC} - 1.5V)$

D.C. CHARACTERISTICS $T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$ (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
I_{CC}	Power Supply Current:				
	8031/8051		175	mA	
	8031AH/8051AH		135	mA	All Outputs
	8032AH/8052AH		175	mA	Disconnected;
	8751H/8751H-8		265	mA	$E_A = V_{CC}$
C_{IO}	Pin Capacitance		10	pF	Test freq = 1 MHz

***NOTE:**

Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the VOLs of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

A.C. CHARACTERISTICS $T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$;

Load Capacitance for Port 0, ALE, and PSEN = 100 pF;

Load Capacitance for All Other Outputs = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL - 40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL - 40		ns
TLLAX	Address Hold after ALE Low	48		TCLCL - 35		ns
TLLIV	ALE Low to Valid Instr In					
	8751H		183		4TCLCL - 150	ns
	All Others		233		4TCLCL - 100	ns
TLLPL	ALE Low to PSEN Low	58		TCLCL - 25		ns
TPLPH	PSEN Pulse Width					
	8751H	190		3TCLCL - 60		ns
	All Others	175		3TCLCL - 75		ns
TPLIV	PSEN Low to Valid Instr In					
	8751H		100		3TCLCL - 150	ns
	All Others		125		3TCLCL - 125	ns
TPXIX	Input Instr Hold after PSEN	0		0		ns
TPXIZ	Input Instr Float after PSEN		63		TCLCL - 20	ns
TPXAV	PSEN to Address Valid	75		TCLCL - 8		ns
TAVIV	Address to Valid Instr In					
	8751H		267		5TCLCL - 150	ns
	All Others		302		5TCLCL - 115	ns
TPLAZ	PSEN Low to Address Float		20		20	ns
TRLRH	RD Pulse Width	400		6TCLCL - 100		ns
TWLWH	WR Pulse Width	400		6TCLCL - 100		ns
TRLDV	RD Low to Valid Data In		252		5TCLCL - 165	ns
TRHDX	Data Hold after RD	0		0		ns
TRHDZ	Data Float after RD		97		2TCLCL - 70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL - 150	ns
TAVDV	Address to Valid Data In		585		9TCLCL - 165	ns
TLLWL	ALE Low to RD or WR Low	200	300	3TCLCL - 50	3TCLCL + 50	ns

A.C. CHARACTERISTICS $T_A = -40^\circ\text{C}$ to $+110^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$;
 Load Capacitance for Port 0, ALE, and PSEN = 100 pF;
 Load Capacitance for All Other Outputs = 80 pF (Continued)

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TAVWL	Address to \overline{RD} or \overline{WR} Low	203		4TCLCL - 130		ns
TQVWX	Data Valid to \overline{WR} Transition					
	8751H	13		TCLCL - 70		ns
	All Others	23		TCLCL - 60		ns
TQVWH	Data Valid to \overline{WR} High	433		7TCLCL - 150		ns
TWHQX	Data Hold after \overline{WR}	33		TCLCL - 50		ns
TRLAZ	\overline{RD} Low to Address Float		20		20	ns
TWHLH	\overline{RD} or \overline{WR} High to ALE High					
	8751H	33	133	TCLCL - 50	TCLCL + 50	ns
	All Others	43	123	TCLCL - 40	TCLCL + 40	ns

NOTE:

*This table does not include the 8751-8 A.C. characteristics (see below).

This Table is only for the 8751H-8

A.C. CHARACTERISTICS $T_A = -40^\circ\text{C}$ to $+110^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$;
 Load Capacitance for Port 0, ALE, and PSEN = 100 pF;
 Load Capacitance for All Other Outputs = 80 pF

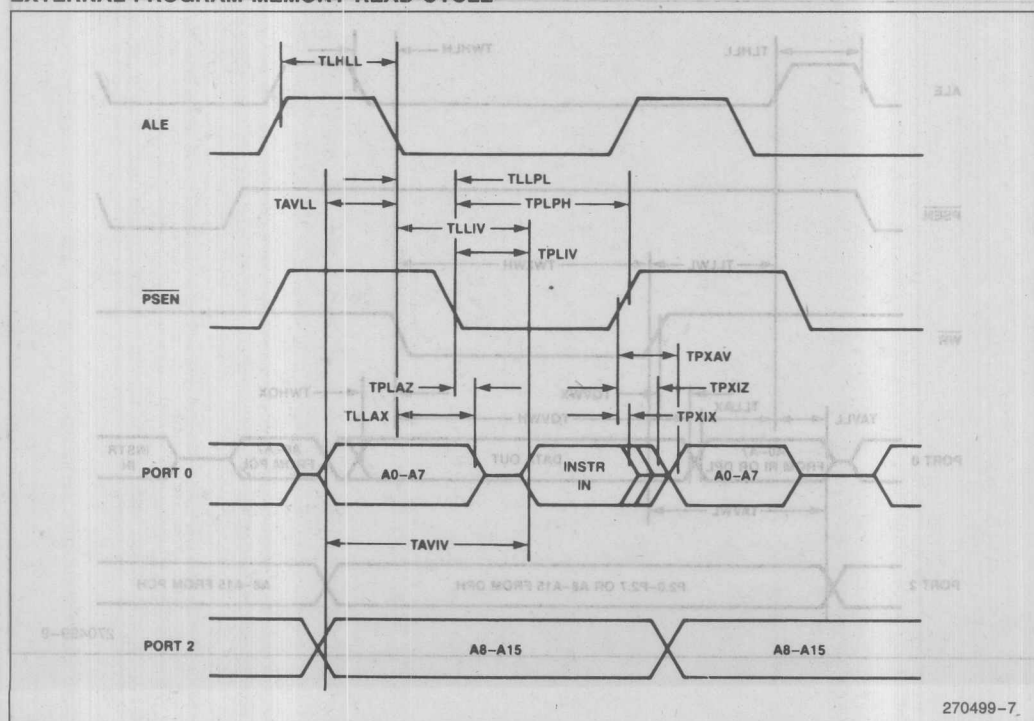
Symbol	Parameter	8 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	8.0	MHz
TLHLL	ALE Pulse Width	210		2TCLCL - 40		ns
TAVLL	Address Valid to ALE Low	85		TCLCL - 40		ns
TLLAX	Address Hold after ALE Low	90		TCLCL - 35		ns
TLLIV	ALE Low to Valid Instr In		350		4TCLCL - 150	ns
TLLPL	ALE Low to \overline{PSEN} Low	100		TCLCL - 25		ns
TPLPH	\overline{PSEN} Pulse Width	315		3TCLCL - 60		ns
TPLIV	\overline{PSEN} Low to Valid Instr In		225		3TCLCL - 150	ns
TPXIX	Input Instr Hold after \overline{PSEN}	0		0		ns
TPXIZ	Input Instr Float after \overline{PSEN}		105		TCLCL - 20	ns
TPXAV	\overline{PSEN} to Address Valid	117		TCLCL - 8		ns
TAVIV	Address to Valid Instr In		475		5TCLCL - 150	ns
TPLAZ	\overline{PSEN} Low to Address Float		20		20	ns
TRLRH	\overline{RD} Pulse Width	650		6TCLCL - 100		ns
TWLWH	\overline{WR} Pulse Width	650		6TCLCL - 100		ns
TRLDV	\overline{RD} Low to Valid Data In		460		5TCLCL - 165	ns
TRHDX	Data Hold after \overline{RD}	0		0		ns

This Table is only for the 8751H-8

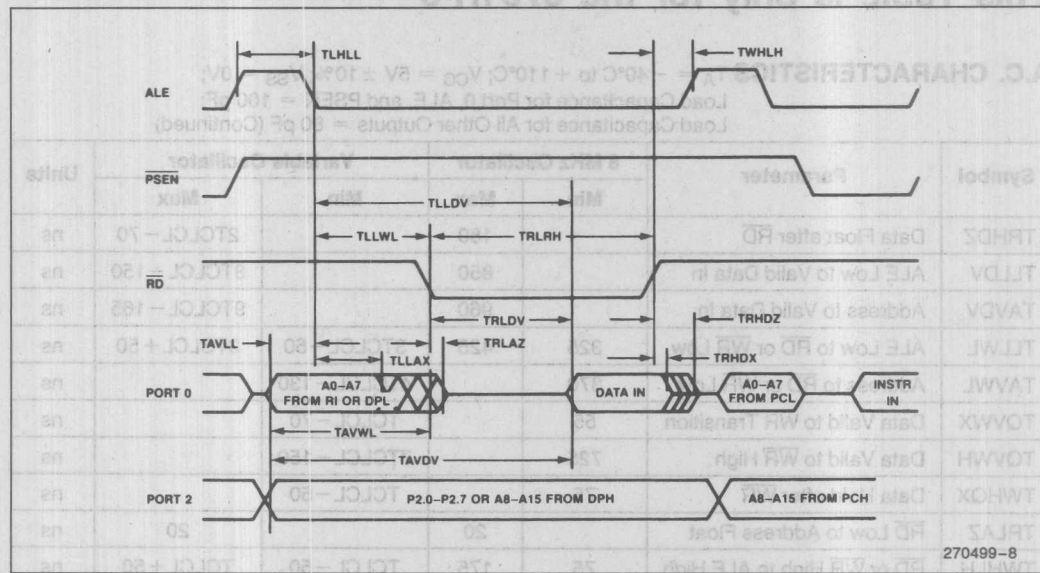
A.C. CHARACTERISTICS $T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$;
 Load Capacitance for Port 0, ALE, and PSEN = 100 pF;
 Load Capacitance for All Other Outputs = 80 pF (Continued)

Symbol	Parameter	8 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TRHDZ	Data Float after $\overline{\text{RD}}$		180		$2\text{TCLCL} - 70$	ns
TLLDV	ALE Low to Valid Data In		850		$8\text{TCLCL} - 150$	ns
TAVDV	Address to Valid Data In		960		$9\text{TCLCL} - 165$	ns
TLLWL	ALE Low to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	325	425	$3\text{TCLCL} - 50$	$3\text{TCLCL} + 50$	ns
TAVWL	Address to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	370		$4\text{TCLCL} - 130$		ns
TQVWX	Data Valid to $\overline{\text{WR}}$ Transition	55		$\text{TCLCL} - 70$		ns
TQVWH	Data Valid to $\overline{\text{WR}}$ High	725		$7\text{TCLCL} - 150$		ns
TWHQX	Data Hold after $\overline{\text{WR}}$	75		$\text{TCLCL} - 50$		ns
TRLAZ	$\overline{\text{RD}}$ Low to Address Float		20		20	ns
TWHLH	$\overline{\text{RD}}$ or $\overline{\text{WR}}$ High to ALE High	75	175	$\text{TCLCL} - 50$	$\text{TCLCL} + 50$	ns

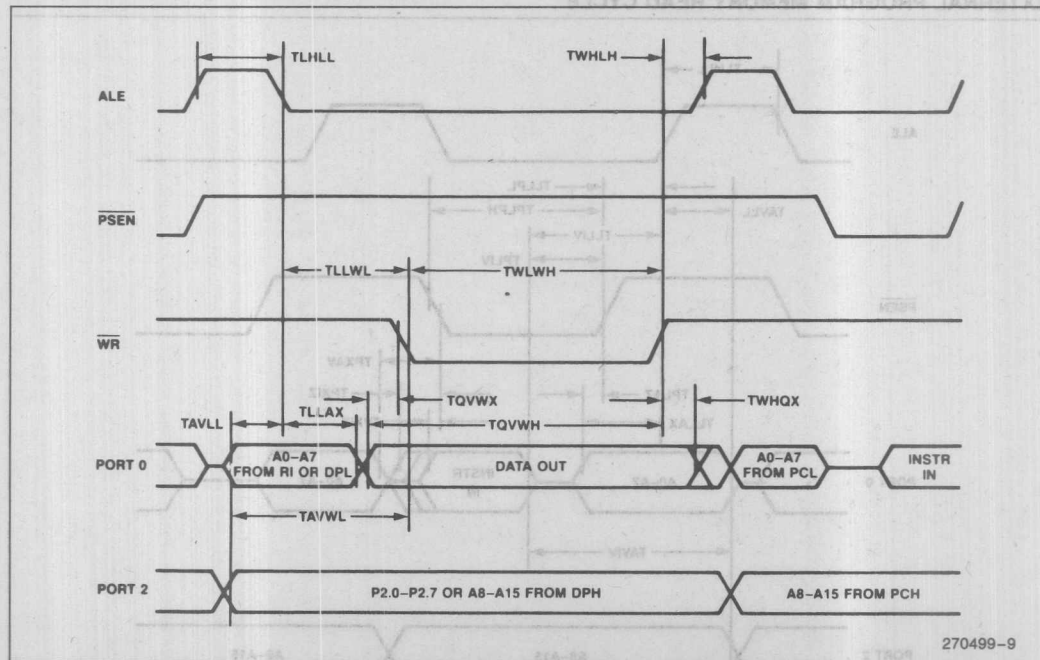
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE

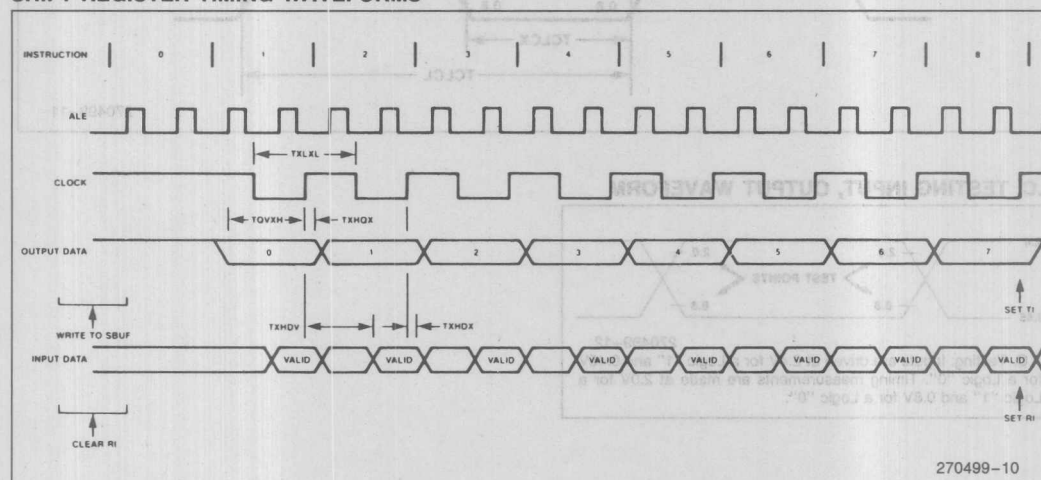


SERIAL PORT TIMING—SHIFT REGISTER MODE

Test Conditions: $T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL - 133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL - 117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL - 133	ns

SHIFT REGISTER TIMING WAVEFORMS

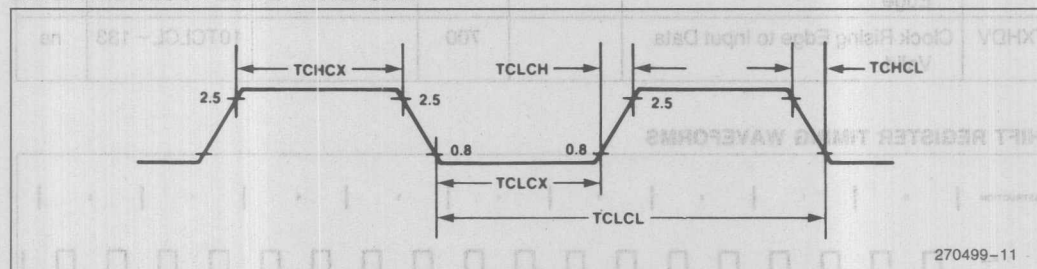


270499-10

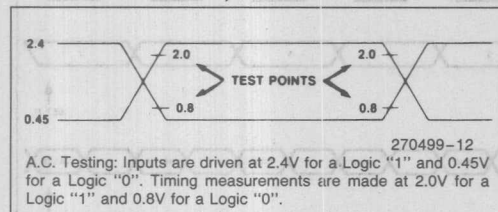
EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency (except 8751H-8)	3.5	12	MHz
	8751H-8	3.5	8	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORM



A.C. TESTING INPUT, OUTPUT WAVEFORM



EPROM CHARACTERISTICS

Table 2. EPROM Programming Modes

Mode	RST	PSEN	ALE	EA	P2.7	P2.6	P2.5	P2.4
Program	1	0	0*	V _{PP}	1	0	X	X
Inhibit	1	0	1	X	1	0	X	X
Verify	1	0	1	1	0	0	X	X
Security Set	1	0	0*	V _{PP}	1	1	X	X

NOTE:

"1" = logic high for that pin
 "0" = logic low for that pin
 "X" = "don't care"

Programming the EPROM

To be programmed, the part must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal registers.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0–P2.3 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 pins, and RST, PSEN, and EA should be held at the "Program" levels indicated in Table 2. ALE is pulsed low for 50 ms to program the code byte into the addressed EPROM location. The setup is shown in Figure 6.

Normally EA is held at a logic high until just before ALE is to be pulsed. Then EA is raised to +21V, ALE is pulsed, and then EA is returned to a logic high. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

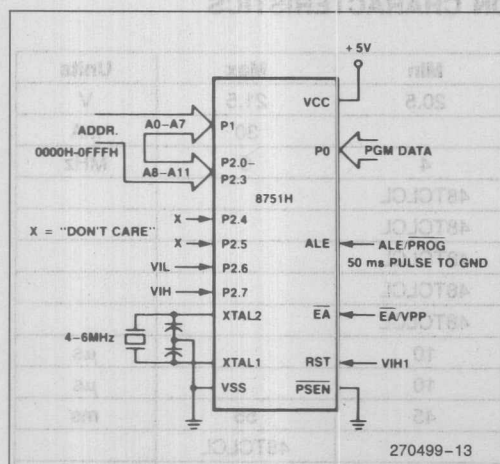


Figure 6. Programming Configuration

"V_{PP}" = +21V ±0.5V

*ALE is pulsed low for 50 ms.

Note that the EA/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level of 21.5V for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches.

Program Verification

If the Security Bit has not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0–P2.3. The other pins should be held at the "Verify" levels indicated in Table 2. The contents of the addressed location will come out on Port 0. External pullups are required on Port 0 for this operation.

The setup, which is shown in Figure 7, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active-low read strobe.

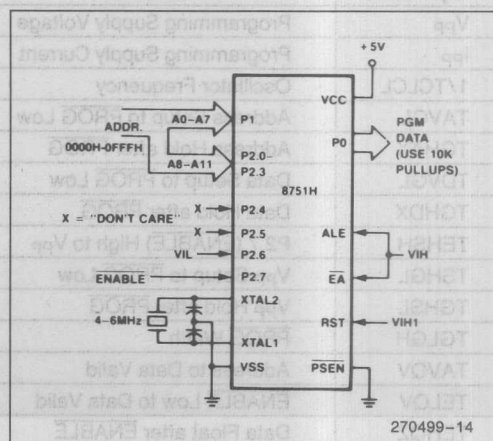


Figure 7. Program Verification

EPROM Security

The security feature consists of a "locking" bit which when programmed denies electrical access by any external means to the on-chip Program Memory. The bit is programmed as shown in Figure 8. The setup and procedure are the same as for normal EPROM programming, except that P2.6 is held at a logic high. Port 0, Port 1, and pins P2.0–P2.3 may be in any state. The other pins should be held at the "Security" levels indicated in Table 2.

Once the Security Bit has been programmed, it can be cleared only by full erasure of the Program Memory. While it is programmed, the internal Program Memory can not be read out, the device can not be further programmed, and it **can not execute out of external program memory**. Erasing the EPROM, thus clearing the Security Bit, restores the device's full functionality. It can then be reprogrammed.

Erase Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

$T_A = 21^\circ\text{C}$ to 27°C ; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	20.5	21.5	V
I_{PP}	Programming Supply Current		30	mA
$1/TCLCL$	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to PROG Low	48TCLCL		
TGHAX	Address Hold after PROG	48TCLCL		
TDVGL	Data Setup to PROG Low	48TCLCL		
TGHDX	Data Hold after PROG	48TCLCL		
TEHSH	P2.7 (ENABLE) High to V_{PP}	48TCLCL		
TSHGL	V_{PP} Setup to PROG Low	10		μs
TGHSL	V_{PP} Hold after PROG	10		μs
TGLGH	PROG Width	45	55	ms
TAVQV	Address to Data Valid		48TCLCL	
TELQV	ENABLE Low to Data Valid		48TCLCL	
TEHQZ	Data Float after ENABLE	0	48TCLCL	

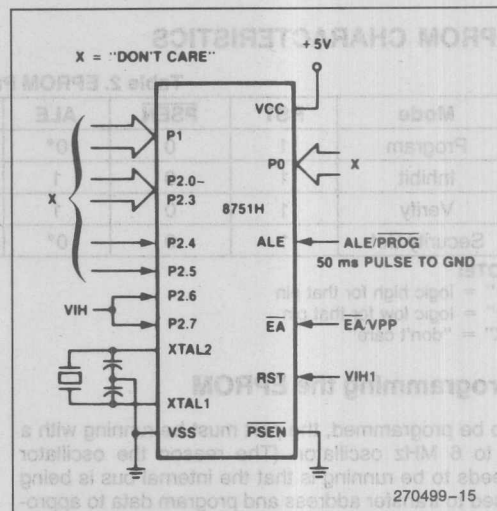
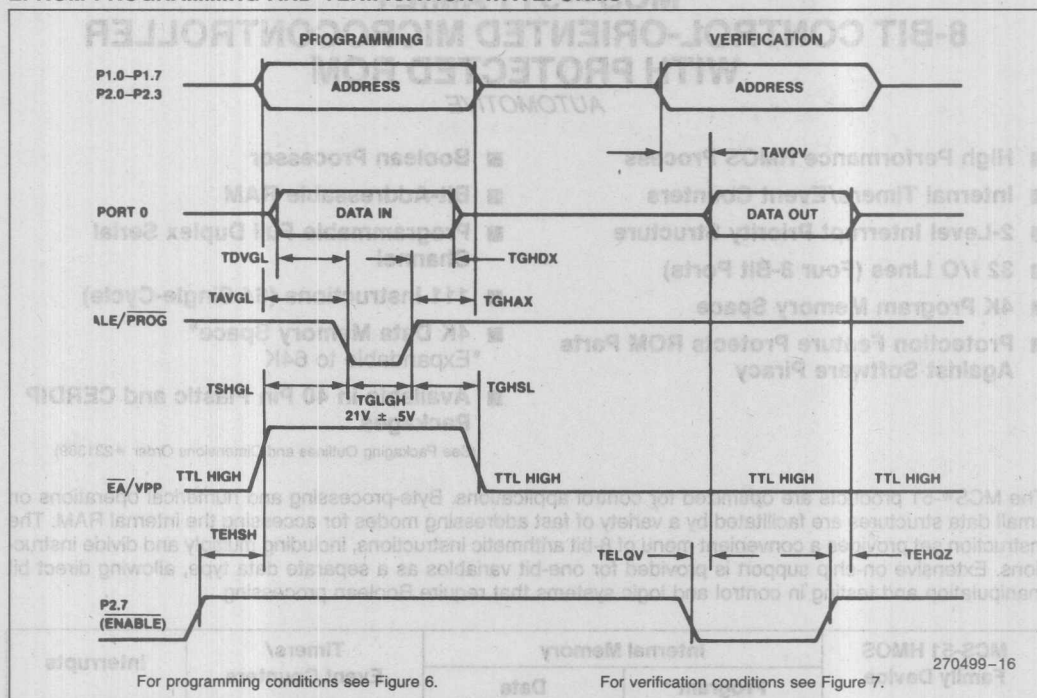


Figure 8. Programming the Security Bit

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm². Exposing the EPROM to an ultraviolet lamp of 12,000 $\mu\text{W}/\text{cm}^2$ rating for 20 to 30 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



The 8051AH is identical to the 8051AH with the exception of the Protection Feature. To incorporate this Protection Feature, program verification has been disabled and external memory accesses have been limited to 4K.



PRELIMINARY

8051AHP
MCS®-51 FAMILY
8-BIT CONTROL-ORIENTED MICROCONTROLLER
WITH PROTECTED ROM
AUTOMOTIVE

- High Performance HMOS Process
- Internal Timers/Event Counters
- 2-Level Interrupt Priority Structure
- 32 I/O Lines (Four 8-Bit Ports)
- 4K Program Memory Space
- Protection Feature Protects ROM Parts Against Software Piracy
- Boolean Processor
- Bit-Addressable RAM
- Programmable Full Duplex Serial Channel
- 111 Instructions (64 Single-Cycle)
- 4K Data Memory Space*
*Expandable to 64K
- Available in 40 Pin Plastic and Cerdip Packages

(See Packaging Outlines and Dimensions Order #231369)

The MCS®-51 products are optimized for control applications. Byte-processing and numerical operations on small data structures are facilitated by a variety of fast addressing modes for accessing the internal RAM. The instruction set provides a convenient menu of 8-bit arithmetic instructions, including multiply and divide instructions. Extensive on-chip support is provided for one-bit variables as a separate data type, allowing direct bit manipulation and testing in control and logic systems that require Boolean processing.

MCS-51 HMOS Family Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051AHP	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5

The 8051AHP is identical to the 8051AH with the exception of the Protection Feature. To incorporate this Protection Feature, program verification has been disabled and external memory accesses have been limited to 4K.

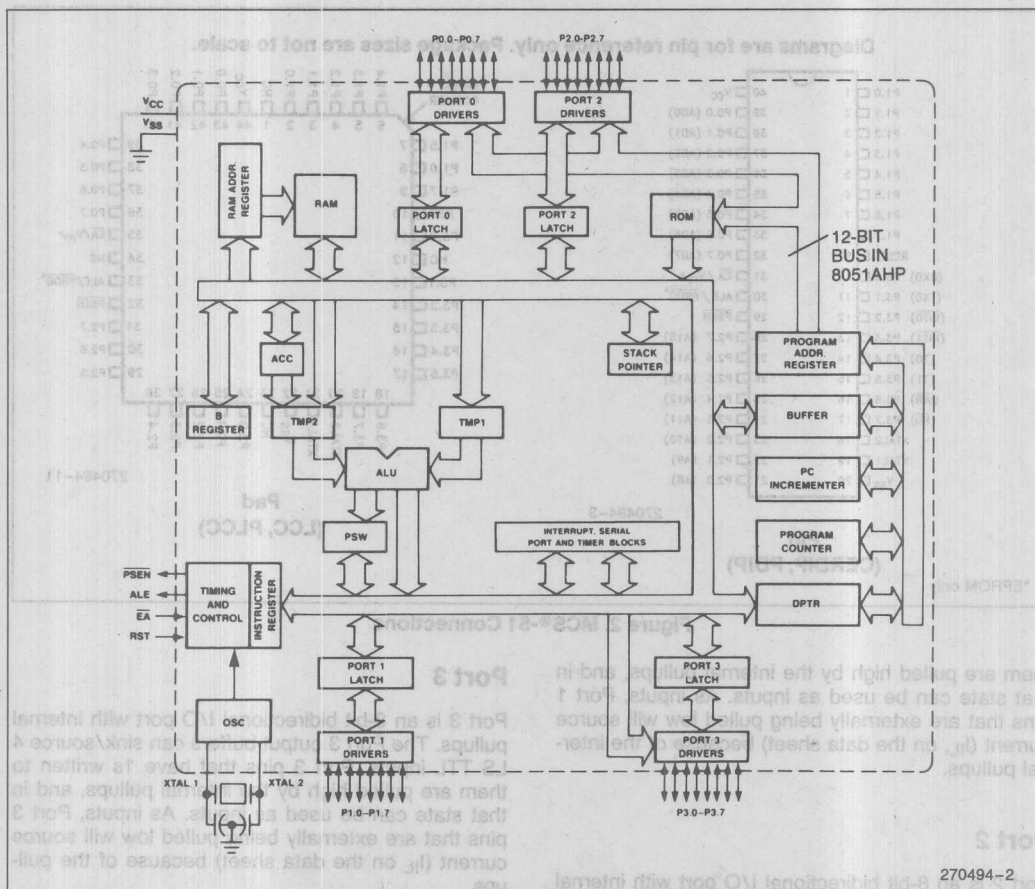


Figure 1. MCS®-51 Block Diagram

PIN DESCRIPTIONS

V_{CC}

Supply voltage.

V_{SS}

Circuit ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs.

Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during programming of the EPROM parts, and outputs the code bytes during program verification of the ROM and EPROM parts. External pullups are required during program verification.

Port 1

Port 1 is an 8 bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink source 4 LS TTL inputs. Port 1 pins that have 1s written to

Diagrams are for pin reference only. Package sizes are not to scale.

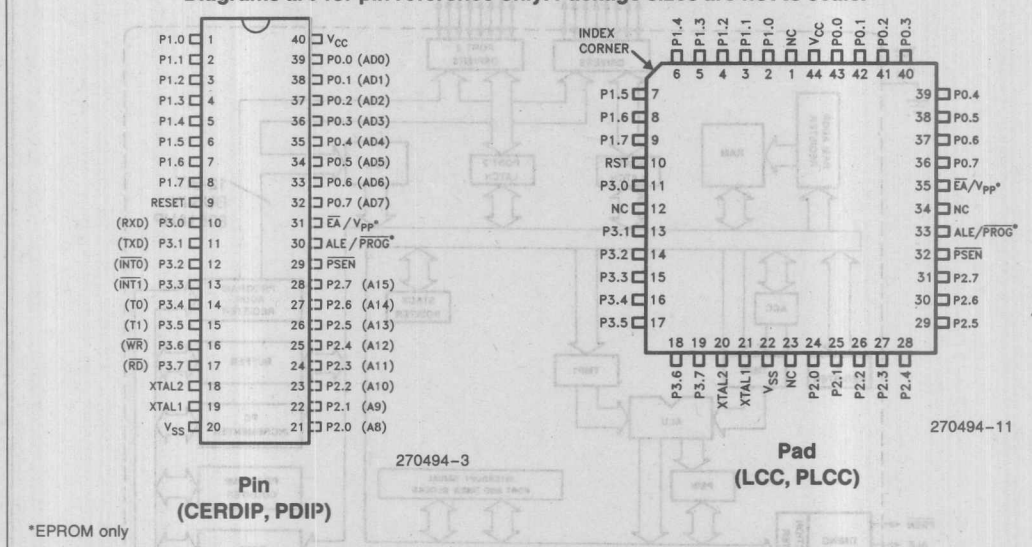


Figure 2. MCS®-51 Connections

them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. Bits P2.4 through P2.7 are forced to 0, effectively limiting external Data and Code space to 4K each in the 8051AHP during external accesses*. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

*Protection feature

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/PROG

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory.

In normal operation ALE is emitted at a constant rate of $\frac{1}{6}$ the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

PSEN

Program Store Enable is the read strobe to external Program Memory.

When the device is executing code from external Program Memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that two $\overline{\text{PSEN}}$ activations are skipped during each access to external Data Memory.

$\overline{\text{EA}}/\text{V}_{\text{PP}}$

External Access enable $\overline{\text{EA}}$ should be strapped to V_{CC} for internal program executions. $\overline{\text{EA}}$ must be strapped to V_{SS} in order to enable any MCS-51 device to fetch code from external Program memory locations 0 to 0FFFH.

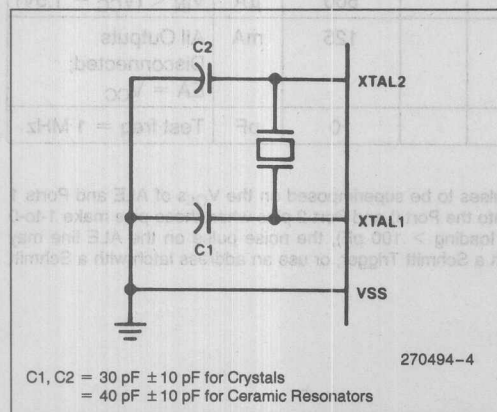


Figure 3. Oscillator Connections

XTAL1

Input to the inverting oscillator amplifier.

XTAL2

Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

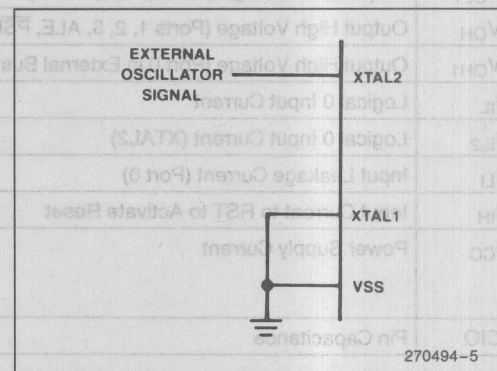


Figure 4. External Drive Configuration

DESIGN CONSIDERATION

The 8051AHP cannot access external Program or Data memory above 4K. This means that the following instructions that use the Data Pointer only read/write data at address locations below 4K:

```
MOVX A, @DPTR
MOVX @DPTR, A
```

When the Data Pointer contains an address above the 4K limit, those locations will not be accessed.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias -40°C to $+110^{\circ}\text{C}$
 Storage Temperature -65°C to $+150^{\circ}\text{C}$
 Voltage on $\overline{\text{EA}}/\text{V}_{\text{PP}}$ Pin to V_{SS} ... -0.5V to $+21.5\text{V}$
 Voltage on Any Other Pin to V_{SS} -0.5V to $+7\text{V}$
 Power Dissipation..... 1.5W

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Typical Thermal Resistance Junction to Ambient

Package
Plastic
CERDIP

θ_{ja}
 75°C/W
 36°C/W

D.C. CHARACTERISTICS $T_{\text{A}} = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $\text{V}_{\text{CC}} = 5\text{V} \pm 10\%$; $\text{V}_{\text{SS}} = 0\text{V}$

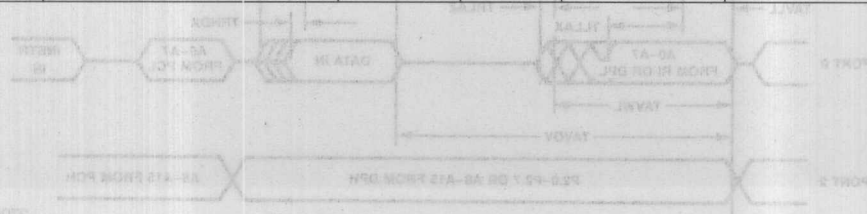
Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage (Except XTAL2, RST)	2.0	$\text{V}_{\text{CC}} + 0.5$	V	
V_{IH1}	Input High Voltage to XTAL2, RST	2.5	$\text{V}_{\text{CC}} + 0.5$	V	$\text{XTAL1} = \text{V}_{\text{SS}}$
V_{OL}	Output Low Voltage (Ports 1, 2, 3)*		0.45	V	$\text{I}_{\text{OL}} = 1.6\text{ mA}$
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{\text{PSEN}}$)*		0.45	V	$\text{I}_{\text{OL}} = 3.2\text{ mA}$
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{\text{PSEN}}$)	2.4		V	$\text{I}_{\text{OH}} = -80\text{ }\mu\text{A}$
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	$\text{I}_{\text{OH}} = -400\text{ }\mu\text{A}$
I_{IL}	Logical 0 Input Current		-500	μA	$\text{V}_{\text{IN}} = 0.45\text{V}$
I_{IL2}	Logical 0 Input Current (XTAL2)		-3.2	mA	$\text{V}_{\text{IN}} = 0.45\text{V}$
I_{LI}	Input Leakage Current (Port 0)		± 10	μA	$0.45 \leq \text{V}_{\text{IN}} \leq \text{V}_{\text{CC}}$
I_{IH}	Input Current to RST to Activate Reset		500	μA	$\text{V}_{\text{IN}} < (\text{V}_{\text{CC}} - 1.5\text{V})$
I_{CC}	Power Supply Current		125	mA	All Outputs Disconnected; $\text{EA} = \text{V}_{\text{CC}}$
CIO	Pin Capacitance		10	pF	Test freq = 1 MHz

*NOTE:

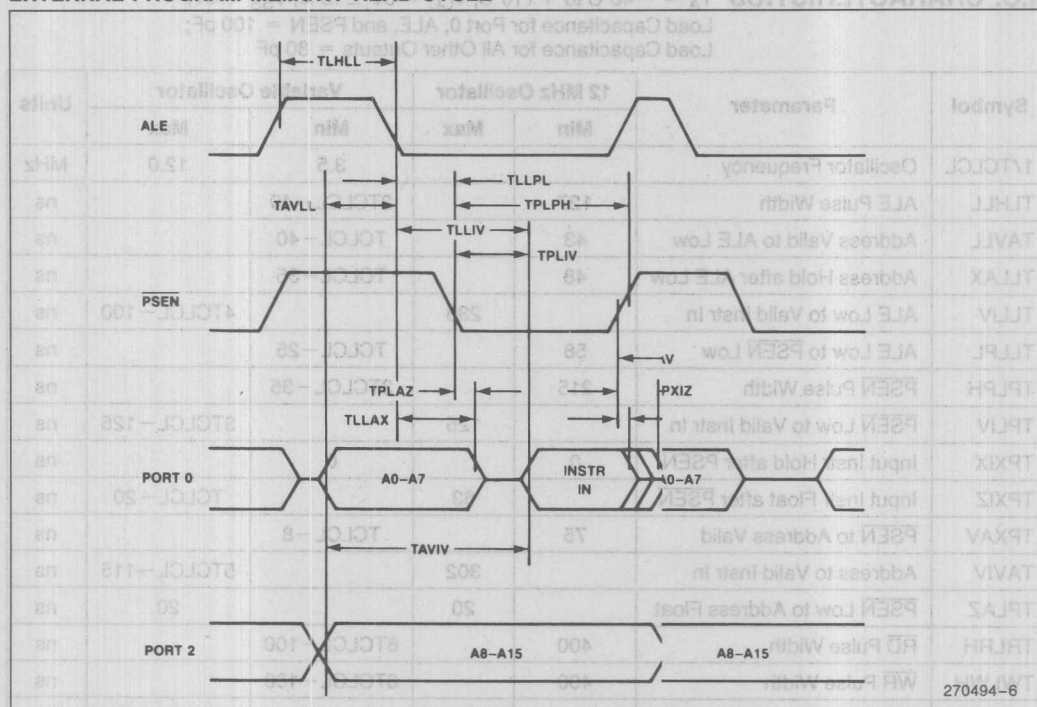
Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL} s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading $> 100\text{ pF}$), the noise pulse on the ALE line may exceed 0.8V . In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

A.C. CHARACTERISTICS $T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$;
Load Capacitance for Port 0, ALE, and PSEN = 100 pF;
Load Capacitance for All Other Outputs = 80 pF

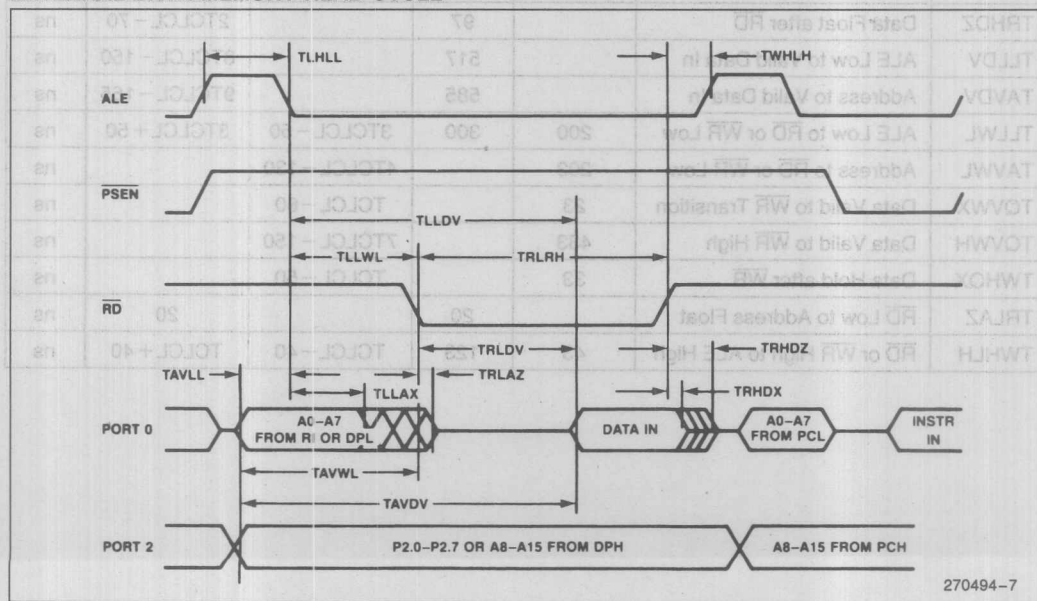
Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL - 40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL - 40		ns
TLLAX	Address Hold after ALE Low	48		TCLCL - 35		ns
TLLIV	ALE Low to Valid Instr In		233		4TCLCL - 100	ns
TLLPL	ALE Low to PSEN Low	58		TCLCL - 25		ns
TPLPH	PSEN Pulse Width	215		3TCLCL - 35		ns
TPLIV	PSEN Low to Valid Instr In		125		3TCLCL - 125	ns
TPXIX	Input Instr Hold after PSEN	0		0		ns
TPXIZ	Input Instr Float after PSEN		63		TCLCL - 20	ns
TPXAV	PSEN to Address Valid	75		TCLCL - 8		ns
TAVIV	Address to Valid Instr In		302		5TCLCL - 115	ns
TPLAZ	PSEN Low to Address Float		20		20	ns
TRLRH	RD Pulse Width	400		6TCLCL - 100		ns
TWLWH	WR Pulse Width	400		6TCLCL - 100		ns
TRLDV	RD Low to Valid Data In		252		5TCLCL - 165	ns
TRHDX	Data Hold after RD	0		0		ns
TRHDZ	Data Float after RD		97		2TCLCL - 70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL - 150	ns
TAVDV	Address to Valid Data In		585		9TCLCL - 165	ns
TLLWL	ALE Low to RD or WR Low	200	300	3TCLCL - 50	3TCLCL + 50	ns
TAVWL	Address to RD or WR Low	203		4TCLCL - 130		ns
TQVWX	Data Valid to WR Transition	23		TCLCL - 60		ns
TQVWH	Data Valid to WR High	433		7TCLCL - 150		ns
TWHQX	Data Hold after WR	33		TCLCL - 50		ns
TRLAZ	RD Low to Address Float		20		20	ns
TWHLH	RD or WR High to ALE High	43	123	TCLCL - 40	TCLCL + 40	ns



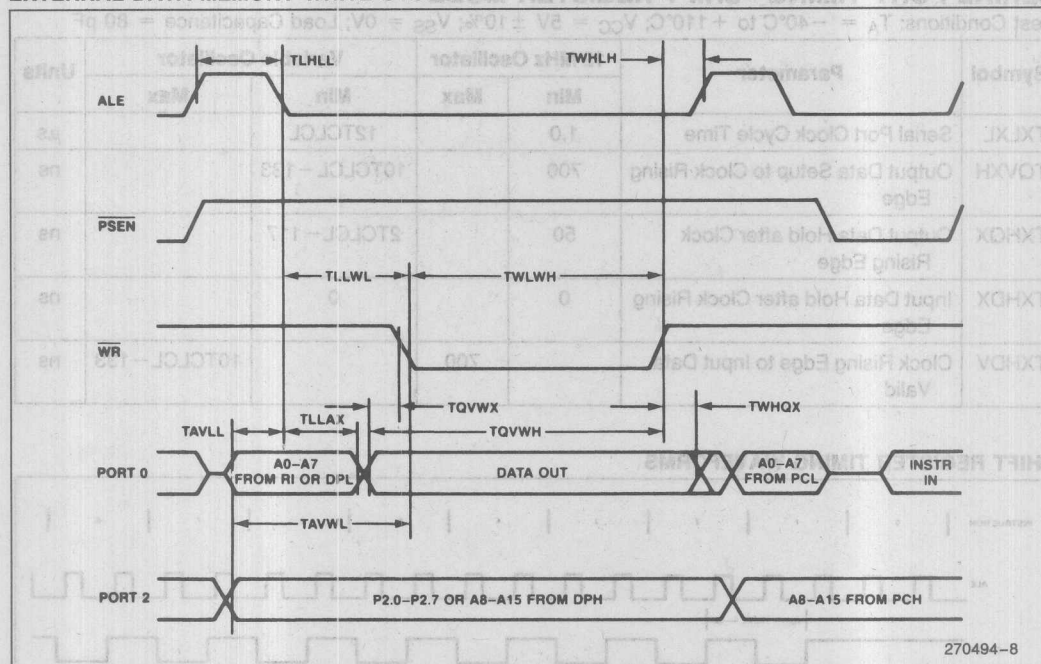
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE

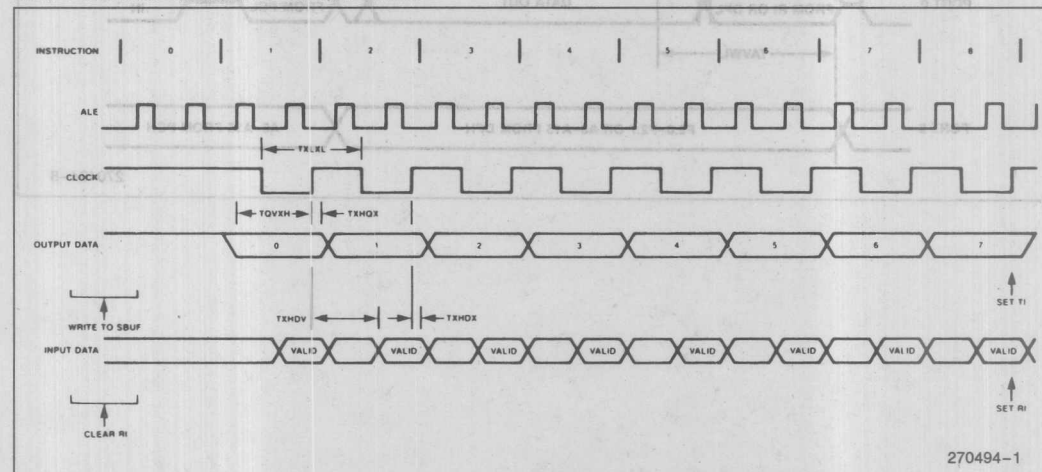


SERIAL PORT TIMING—SHIFT REGISTER MODE

Test Conditions: $T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL – 133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL – 117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL – 133	ns

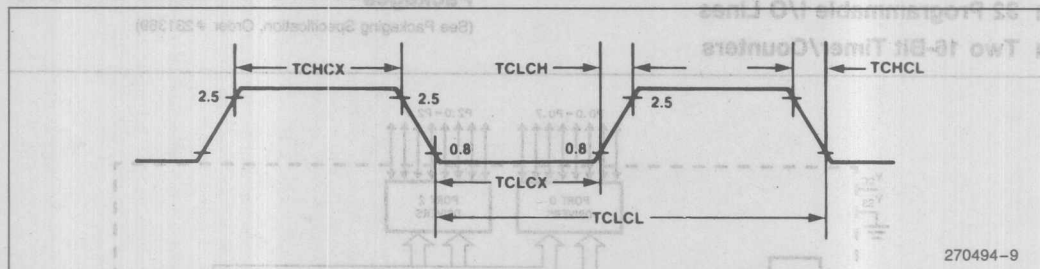
SHIFT REGISTER TIMING WAVEFORMS



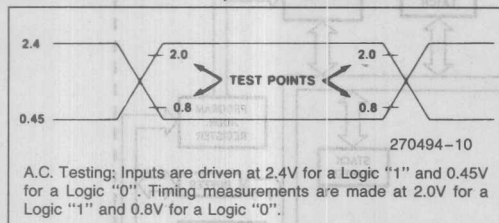
EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORM

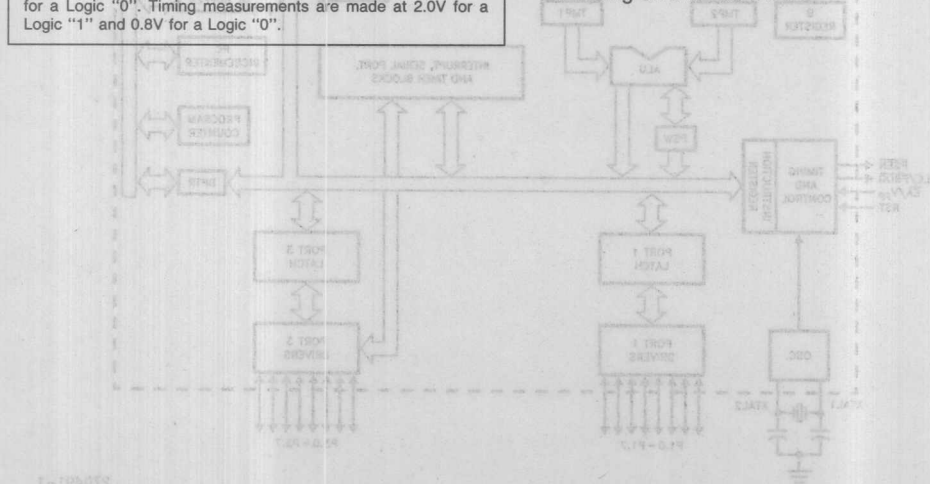


A.C. TESTING INPUT, OUTPUT WAVEFORM



Program Verification

The program verification test mode has been eliminated on the 8051AHP. It is not possible to verify the ROM contents using this mode, the way EPROM programmers typically do. Also, the ROM contents cannot be verified by a program executing out of external program memory due to the restricted addressing on the 8051AHP.



SINGLE-CHIP 8-BIT MICROCOMPUTER WITH 4K BYTES OF EPROM PROGRAM MEMORY

Automotive

- Program Memory Lock
 - 128 Bytes Data Ram
 - Quick Pulse Programming™ Algorithm
 - 12.75 Volt Programming Voltage
 - Boolean Processor
 - 32 Programmable I/O Lines
 - Two 16-Bit Timer/Counters
 - 5 Interrupt Sources
 - Programmable Serial Channel
 - 64K External Program Memory Space
 - 64K External Data Memory Space
 - Available in LCC, PLCC and DIP Packages
- (See Packaging Specification, Order #231369)

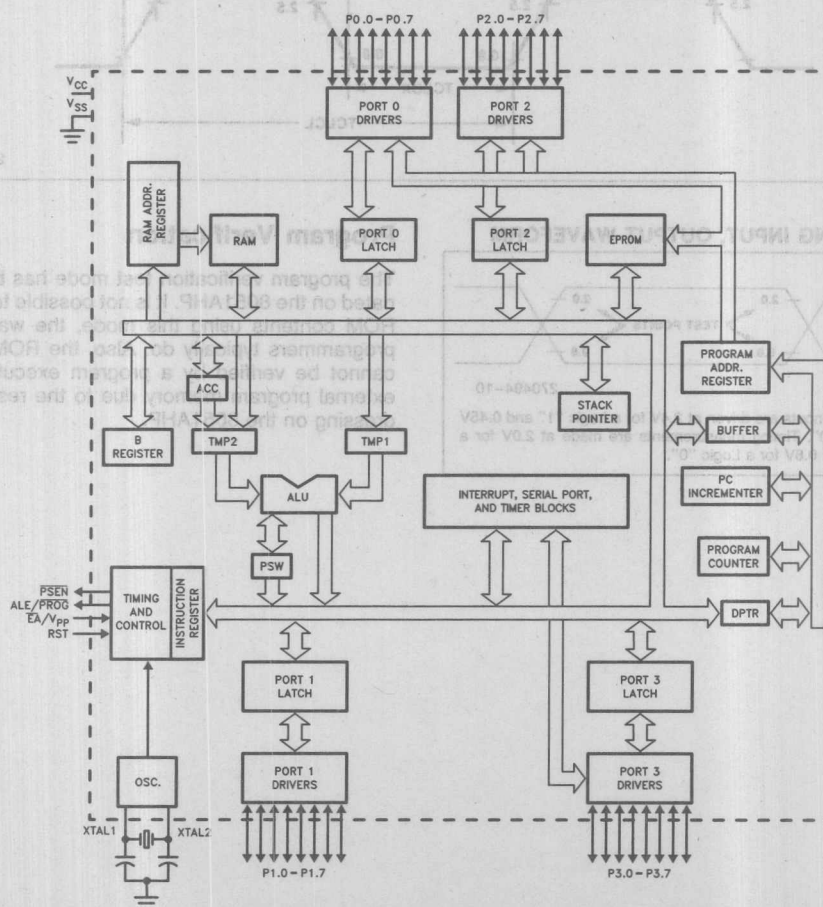


Figure 1. 8751BH Block Diagram

270491-1

8751BH PRODUCT OPTIONS

Intel's extended and automotive temperature range products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to 70°C ambient. With the extended temperature range option, operational characteristics are guaranteed over the temperature

range of -40°C to +85°C ambient. For the automotive temperature range option, operational characteristics are guaranteed over the temperature range of -40°C to +110°C ambient.

The automotive, extended, and commercial temperature versions of the MCS®-51 product families are available with or without burn-in options as listed in Table 1.

As shown in Figure 2, temperature, burn-in, and package options are identified by a one- or two-letter prefix to the part number.

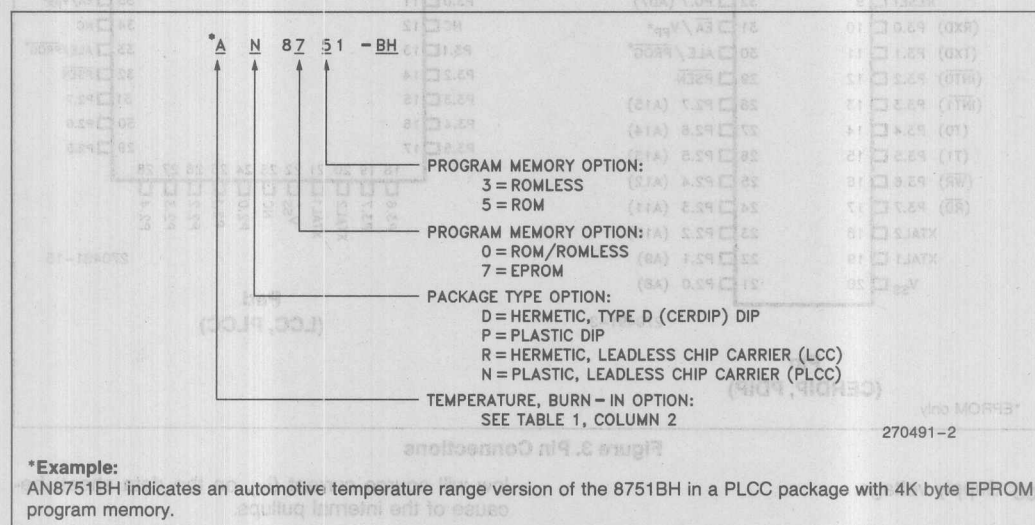


Figure 2. MCS®-51 Product Family Nomenclature

Table 1. Temperature-Burn-In Options

Temperature Classification	Temperature Designation	Operating Temperature °C Ambient	Burn-In 125°C (Hr)
Commercial	Null	0 to 70	None
	Q	0 to 70	168 ± 8
Extended	T	-40 to +85	None
	L	-40 to +85	168 ± 8
Automotive	A	-40 to +110°C	None
	B	-40 to +110°C	168 ± 8

NOTE:

Other Burn-In options are also available, but not standard.

PIN DESCRIPTIONS

Diagrams are for pin reference only. Package sizes are not to scale.

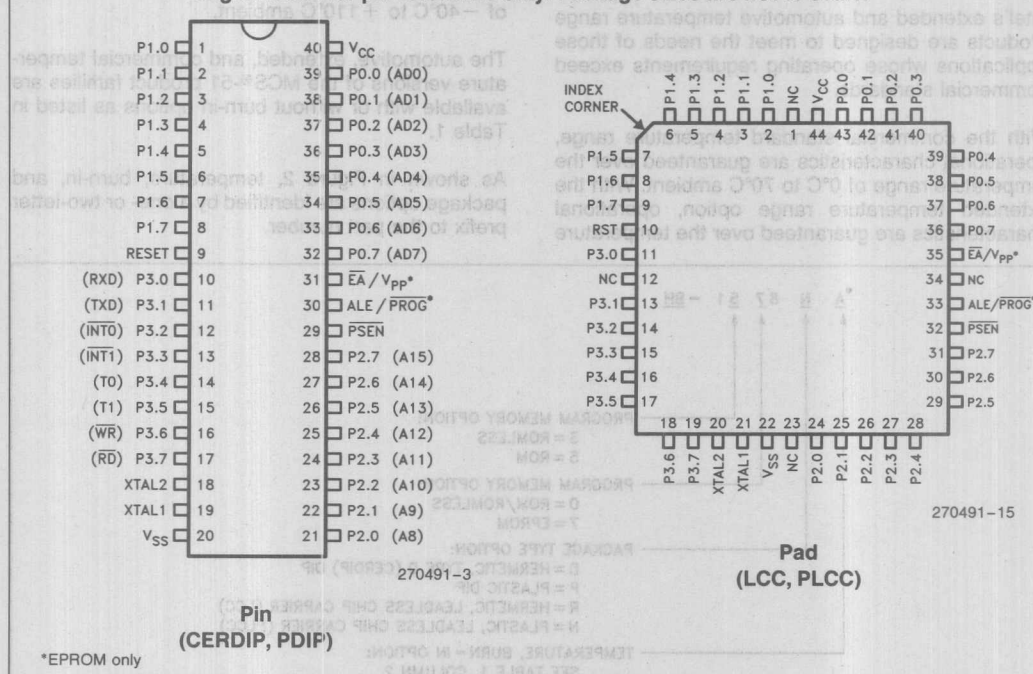


Figure 3. Pin Connections

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s, and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during EPROM programming, and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled

low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during EPROM programming and program verification.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during EPROM programming and program verification.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} , on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	\overline{WR} (external data memory write strobe)
P3.7	\overline{RD} (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/PROG: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during EPROM programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

PSEN: Program Store Enable is the Read strobe to External Program Memory.

When the 8751BH is executing code from external Program Memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to External Data Memory.

\overline{EA}/V_{PP} : External Access enable. \overline{EA} must be strapped to V_{SS} in order to enable the device to fetch code from External Program Memory locations 0000H to 0FFFH. Note, however, that if either of the Lock Bits are programmed, \overline{EA} will be internally latched on reset.

\overline{EA} should be strapped to V_{CC} for internal program executions.

This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 4. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Applications Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 5. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

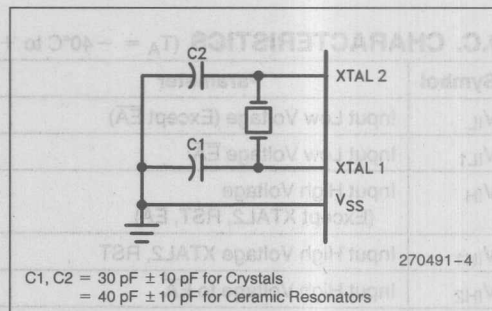


Figure 4. Oscillator Connections

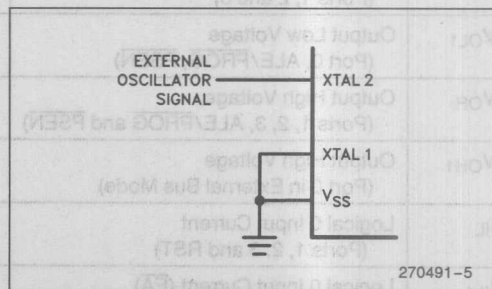


Figure 5. External Clock Drive Configuration

DESIGN CONSIDERATIONS

Exposure to light when the device is in operation may cause logic errors. For this reason, it is suggested that an opaque label be placed over the window when the die is exposed to ambient light.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature	Under Bias	−40°C to +110°C
Storage Temperature		−65°C to +150°C
Voltage on \overline{EA}/V_{PP} Pin to V_{SS}		−0.5V to +13.0V
Voltage on Any Other Pin to V_{SS}		−0.5V to +7V
Power Dissipation		1.5W
(based on PACKAGE heat transfer limitations, not device power consumption)		

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

Typical Thermal Resistance Junction to Ambient (θ_{JA})

Package	θ_{JA}
CERDIP	36°C/W
CERAMIC	34°C/W
PLCC	45.6°C/W

ADVANCE INFORMATION—SEE INTEL FOR DESIGN-IN INFORMATION

D.C. CHARACTERISTICS ($T_A = -40^\circ\text{C}$ to $+110^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except \overline{EA})	−0.5	0.8	V	
V_{IL1}	Input Low Voltage \overline{EA}	V_{SS}	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST, \overline{EA})	2.0	$V_{CC} + 0.5$	V	
V_{IH1}	Input High Voltage XTAL2, RST	2.5	$V_{CC} + 0.5$	V	XTAL1 = V_{SS}
V_{IH2}	Input High Voltage to \overline{EA}	4.5	5.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2 and 3)		0.45	V	$I_{OL} = 1.6\text{ mA}$ (Note 1)
V_{OL1}	Output Low Voltage (Port 0, ALE/ \overline{PROG} , \overline{PSEN})		0.45	V	$I_{OL} = 3.2\text{ mA}$ (Notes 1, 2)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE/ \overline{PROG} and \overline{PSEN})	2.4		V	$I_{OH} = -80\text{ }\mu\text{A}$
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	$I_{OH} = -400\text{ }\mu\text{A}$
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3 and RST)		−1	mA	$V_{IN} = 0.45V$
I_{IL1}	Logical 0 Input Current (\overline{EA})		−10	mA	$V_{IN} = V_{SS}$
I_{IL2}	Logical 0 Input Current (XTAL2)		−3.2	mA	$V_{IN} = 0.45V$ XTAL1 = V_{SS}
I_{LI}	Input Leakage Current (Port 0)		± 10	μA	$0.45 < V_{IN} < V_{CC}$
I_{IH}	Logical 1 Input Current (\overline{EA})		1	mA	$4.5V < V_{IN} < 5.5V$
I_{IH1}	Input Current to RST to Activate Reset		500	μA	$V_{IN} < (V_{CC} - 1.5V)$

D.C. CHARACTERISTICS ($T_A = -40^\circ\text{C}$ to $+110^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$) (Continued)

Symbol	Parameter	Min	Max	Unit	Test Conditions
I_{CC}	Power Supply Current		175	mA	All Outputs Disconnected
C_{IO}	Pin Capacitance		10	pF	Test Freq = 1MHz

NOTES:

1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OLS} of ALE/PROG and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading $> 100\text{pF}$), the noise pulse on the ALE/PROG pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

2. ALE/PROG refers to a pin on the 8751BH. ALE refers to a timing signal that is output on the ALE/PROG pin.

A.C. CHARACTERISTICS ($T_A = -40^\circ\text{C}$ to $+110^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$); Load Capacitance for Port 0, ALE/PROG, and PSEN = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

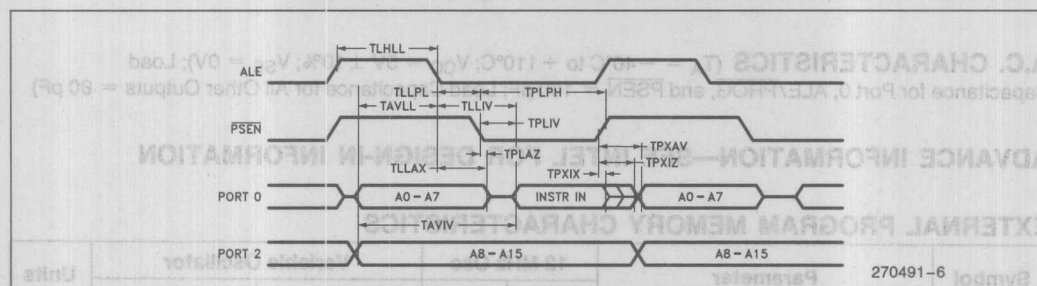
ADVANCE INFORMATION—SEE INTEL FOR DESIGN-IN INFORMATION

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

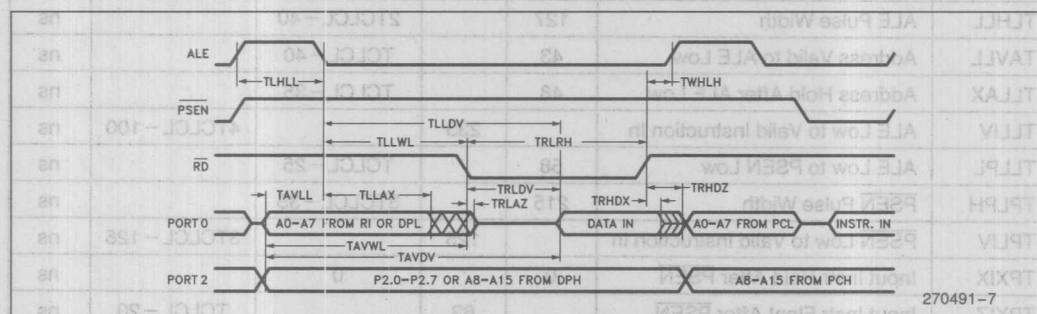
Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL - 40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL - 40		ns
TLLAX	Address Hold After ALE Low	48		TCLCL - 35		ns
TLLIV	ALE Low to Valid Instruction In		233		4TCLCL - 100	ns
TLLPL	ALE Low to PSEN Low	58		TCLCL - 25		ns
TPLPH	PSEN Pulse Width	215		3TCLCL - 35		ns
TPLIV	PSEN Low to Valid Instruction In		125		3TCLCL - 125	ns
TPXIX	Input Instr Hold After PSEN	0		0		ns
TPXIZ	Input Instr Float After PSEN		63		TCLCL - 20	ns
TPXAV	PSEN to Address Valid	75		TCLCL - 8		ns
TAVIV	Address to Valid Instruction In		302		5TCLCL - 115	ns
TPLAZ	PSEN Low to Address Float		20		20	ns
TRLRH	RD Pulse Width	400		6TCLCL - 100		ns
TWLWH	WR Pulse Width	400		6TCLCL - 100		ns
TRLDV	RD Low to Valid Data In		252		5TCLCL - 165	ns
TRHDX	Data Hold After RD	0		0		ns
TRHDZ	Data Float After RD		97		2TCLCL - 70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL - 150	ns
TAVDV	Address to Valid Data In		585		9TCLCL - 165	ns
TLLWL	ALE Low to RD or WR Low	200	300	3TCLCL - 50	3TCLCL + 50	ns
TAVWL	Address to RD or WR Low	203		4TCLCL - 130		ns

EXTERNAL PROGRAM MEMORY CHARACTERISTICS (Continued)

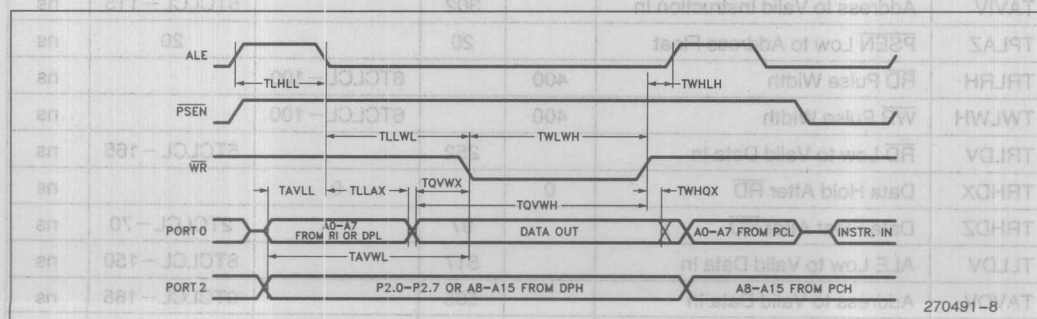
Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TQVWX	Data Valid to \overline{WR} Transition	23		TCLCL - 60		ns
TQVWH	Data Valid to \overline{WR} High	433		7TCLCL - 150		ns
TWHQX	Data Held After \overline{WR}	33		TCLCL - 50		ns
TRLAZ	\overline{RD} Low to Address Float		0		0	ns
TWHLH	\overline{RD} or \overline{WR} High to ALE High	43	123	TCLCL - 40	TCLCL + 40	ns



External Program Memory Read Cycle



External Data Memory Read Cycle



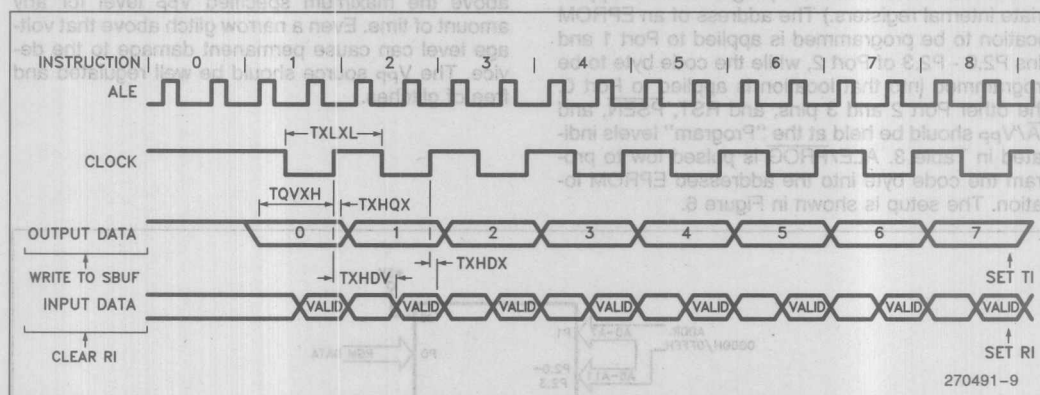
External Data Memory Write Cycle

SERIAL PORT TIMING — SHIFT REGISTER MODE

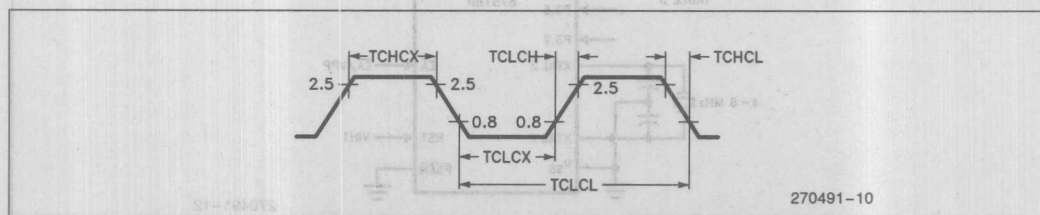
TEST CONDITIONS

($T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL - 133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL - 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700	10TCLCL - 133		ns



Shift Register Mode Timing Waveforms



External Clock Drive Waveforms

EXTERNAL CLOCK DRIVE

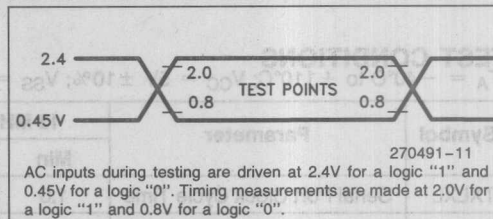
Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EPROM CHARACTERISTICS

Programming the EPROM

To be programmed, the part must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal registers.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0 - P2.3 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 and 3 pins, and RST, PSEN, and EA/V_{PP} should be held at the "Program" levels indicated in Table 3. ALE/PROG is pulsed low to program the code byte into the addressed EPROM location. The setup is shown in Figure 6.

AC TESTING INPUT/OUTPUT WAVEFORMS



Normally EA/V_{PP} is held at a logic high until just before ALE/PROG is to be pulsed. Then EA/V_{PP} is raised to V_{PP}, ALE/PROG is pulsed low, and then EA/V_{PP} is returned to a valid high voltage. The voltage on the EA/V_{PP} pin must be at the valid EA/V_{PP} high level before a verify is attempted. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

Note that the EA/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches.

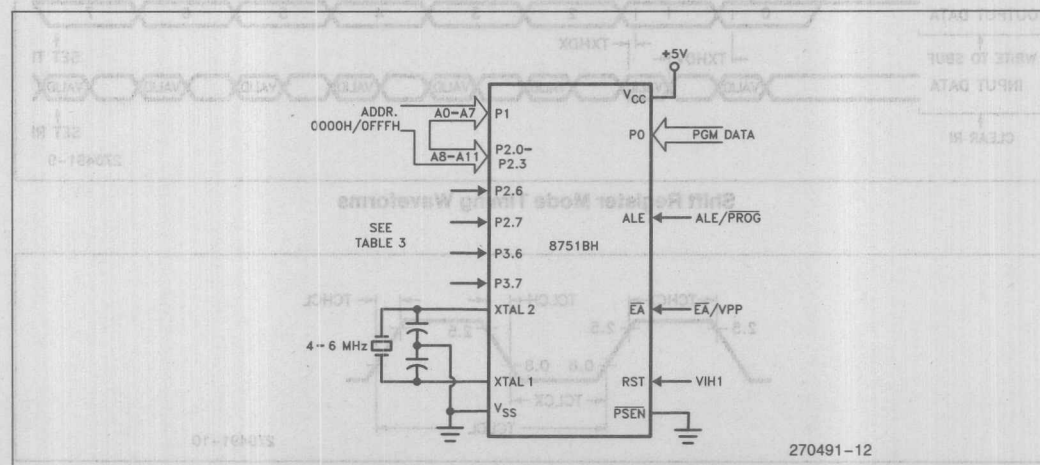


Figure 6. Programming the EPROM

Table 2. EPROM Programming Modes

MODE	RST	PSEN	ALE/ PROG	$\overline{\text{EA}}/\text{V}_{\text{PP}}$	P2.7	P2.6	P3.6	P3.7
Program Code Data	1	0	0*	V _{PP}	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Program Encryption Table Use Addresses 0-1FH	1	0	0*	V _{PP}	1	0	0	1
Program Lock x = 1	1	0	0*	V _{PP}	1	1	1	1
Bits (LBx) x = 2	1	0	0*	V _{PP}	1	1	0	0

NOTES:

"1" = Valid high for that pin

"0" = Valid low for that pin

"V_{PP}" = +12.75V ± 0.25V

* ALE/PROG is pulsed low for 100 μ s for programming. (Quick-Pulse Programming)

QUICK-PULSE PROGRAMMING™ ALGORITHM

The 8751BH can be programmed using the Quick-Pulse Programming Algorithm for microcontrollers. The features of the new programming method are a lower V_{pp} (12.75 volts as compared to 21 volts) and a shorter programming pulse. It is possible to program the entire 4K Bytes of EPROM memory in less than 13 seconds with this algorithm.

To program the part using the new algorithm, V_{PP} must be 12.75 ± 0.25 Volts. ALE/PROG is pulsed low for 100 μ seconds, 25 times. Then, the byte just programmed may be verified. After programming, the entire array should be verified. The Program Lock features are programmed using the same method, but with the setup as shown in Table 2. The only difference in programming Lock features is that the Lock features cannot be directly verified. Instead, verification of programming is by observing that their features are enabled.

PROGRAM VERIFICATION

If the Lock Bits have not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0 - P2.3. The other pins should be held at the "Verify" levels indicated in Table 2. The con-

tents of the addressed location will come out on Port 0. External pullups are required on Port 0 for this operation. (If the Encryption Array in the EPROM has been programmed, the data present at Port 0 will be Code Data XOR Encryption Data. The user must know the Encryption Array contents to manually "unencrypt" the data during verify.)

The setup, which is shown in Figure 7, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active low read strobe.

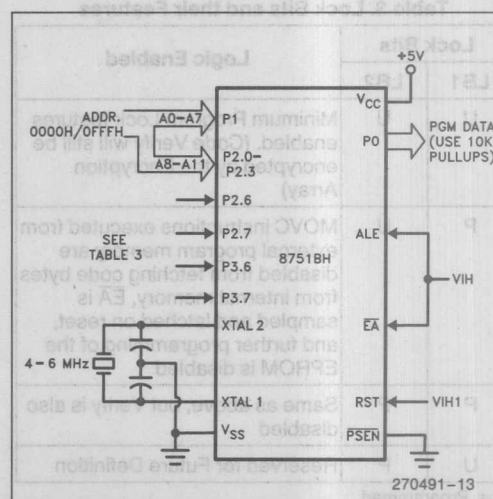


Figure 7. Verifying the EPROM

PROGRAM MEMORY LOCK

The two-level Program Lock system consists of 2 Lock bits and a 32-byte Encryption Array which are used to protect the program memory against software piracy.

ENCRYPTION ARRAY

Within the EPROM array are 32 bytes of Encryption Array that are initially unprogrammed (all 1s). Every time that a byte is addressed during a verify, 5 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NORed (XNOR) with the code byte, creating an Encrypted Verify byte. The algorithm, with the array in the unprogrammed state (all 1s), will return the code in its original, unmodified form.

It is recommended that whenever the Encryption Array is used, at least one of the Lock Bits be programmed as well.

LOCK BITS

Also included in the EPROM Program Lock scheme are two Lock Bits which function as shown in Table 3.

Table 3. Lock Bits and their Features

Lock Bits		Logic Enabled
LB1	LB2	
U	U	Minimum Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array)
P	U	MOVX instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled
P	P	Same as above, but Verify is also disabled
U	P	Reserved for Future Definition

P = Programmed
U = Unprogrammed

Erasing the EPROM also erases the Encryption Array and the Lock Bits, returning the part to full unlocked functionality.

To ensure proper functionality of the chip, the internally latched value of the EA pin must agree with its external state.

ERASURE CHARACTERISTICS

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm. Exposing the EPROM to an ultraviolet lamp of 12,000 μ W/cm rating for 20 to 30 minutes, at a distance of about 1 inch, should be sufficient.

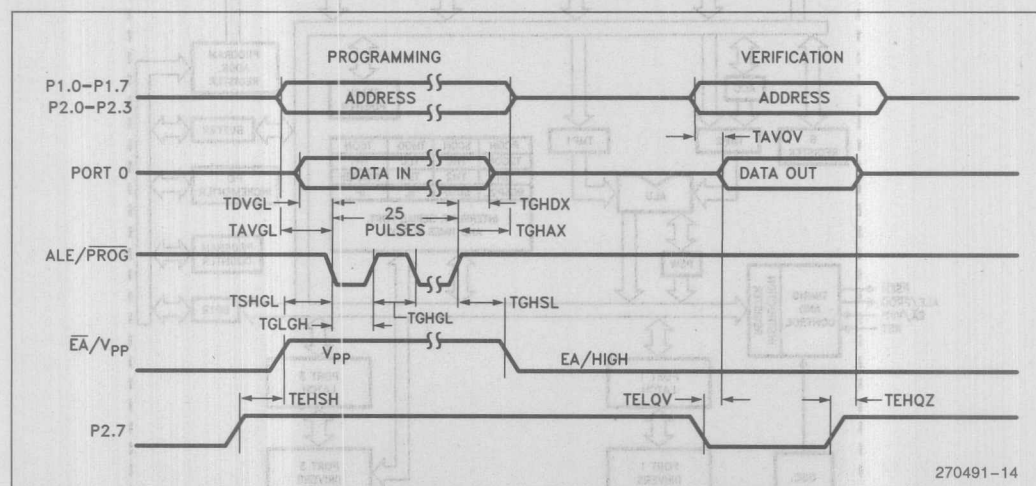
Erasure leaves the array in an all 1s state.

ADVANCE INFORMATION—SEE INTEL FOR DESIGN-IN INFORMATION

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

($T_A = 21^\circ\text{C}$ to 27°C , $V_{CC} = 5.0\text{V} \pm 10\%$, $V_{SS} = 0\text{V}$)

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	12.5	13.0	V
IPP	Programming Supply Current		50	mA
$1/\text{TCLCL}$	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHAX	Address Hold After $\overline{\text{PROG}}$	48TCLCL		
TDVGL	Data Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHDX	Data Hold After $\overline{\text{PROG}}$	48TCLCL		
TEHSH	P2.7 ($\overline{\text{ENABLE}}$) High to V_{PP}	48TCLCL		
TSHGL	V_{PP} Setup to $\overline{\text{PROG}}$ Low	10		μsec
TGHSL	V_{PP} Hold After $\overline{\text{PROG}}$	10		μsec
TGLGH	$\overline{\text{PROG}}$ Width	90	110	μsec
TAVQV	Address to Data Valid		48TCLCL	
TELQV	ENABLE Low to Data Valid		48TCLCL	
TEHQZ	Data Float After ENABLE	0	48TCLCL	
TGHGL	$\overline{\text{PROG}}$ High to $\overline{\text{PROG}}$ Low	10		μsec



EPROM Programming and Verification Waveforms

270491-14

8052BH
SINGLE-CHIP 8-BIT MICROCOMPUTER
WITH FACTORY MASK-PROGRAMMABLE ROM
8032BH
SINGLE-CHIP 8-BIT CONTROL-ORIENTED
CPU WITH RAM AND I/O

AUTOMOTIVE

- 256 Bytes Data Ram
- Boolean Processor
- 32 Programmable I/O Lines
- Three 16-Bit Timer/Counters
- 6 Interrupt Sources
- Programmable Serial Channel
- Separate Transmit/Receive Baud Rate Capability
- 64K External Program Memory Space
- 64K External Data Memory Space
- Available in LCC, PLCC and DIP Packages

(See Packaging Specifications, Order #231369)

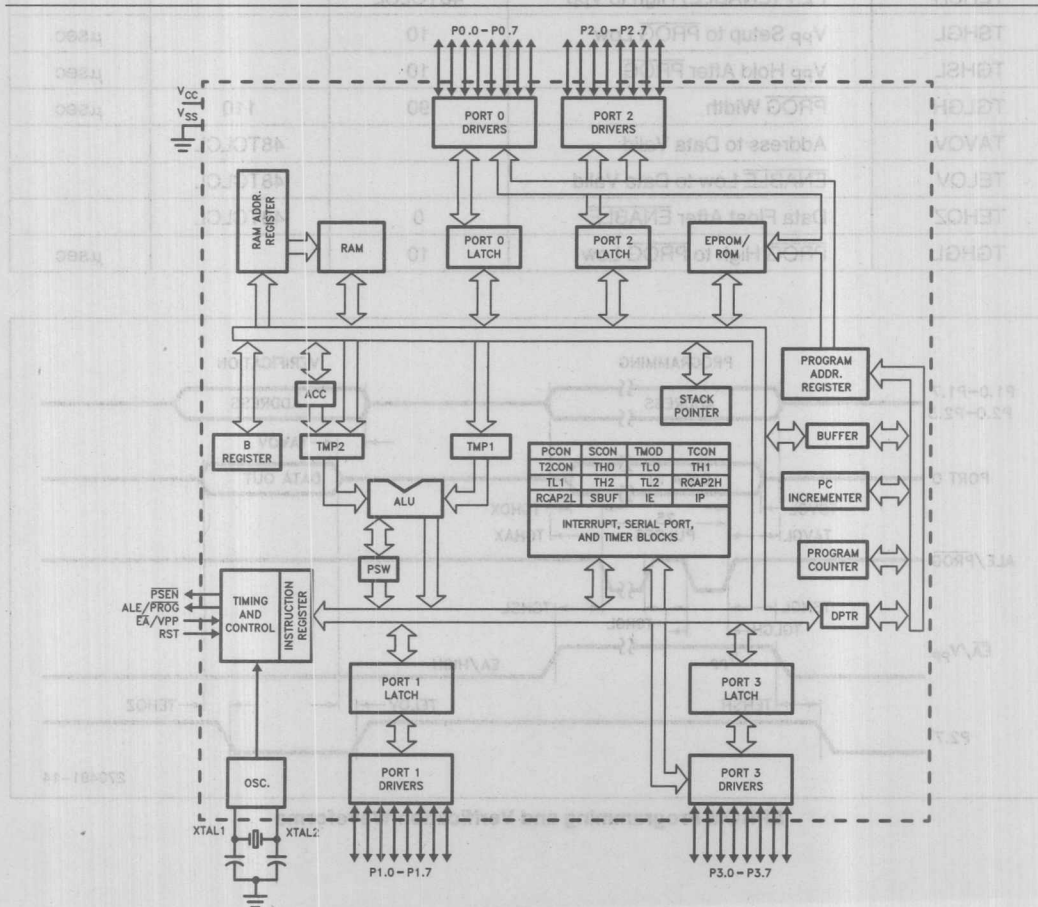


Figure 1. Block Diagram

270496-1

8052BH/8032BH PRODUCT OPTIONS

Intel's extended and automotive temperature range products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to 70°C ambient. With the extended temperature range option, operational characteristics are guaranteed over the temperature

range of -40°C to +85°C ambient. For the automotive temperature range option, operational characteristics are guaranteed over the temperature range of -40°C to +110°C ambient.

The automotive, extended, and commercial temperature versions of the MCS-51 product families are available with or without burn-in options as listed in Table 1.

As shown in Figure 2, temperature, burn-in, and package options are identified by a one- or two-letter prefix to the part number.

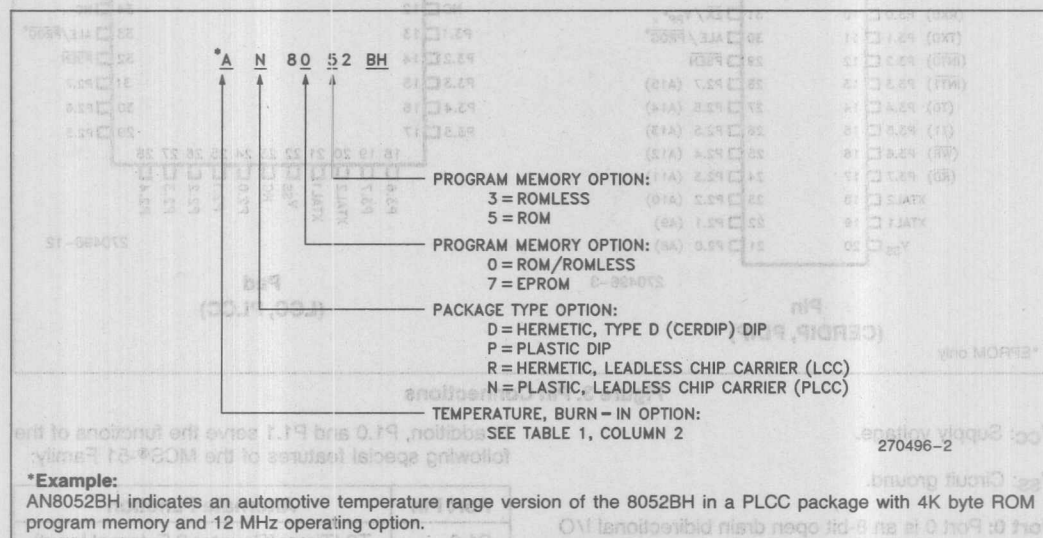


Figure 2. MCS®-51 Product Family Nomenclature

Table 1. Temperature-Burn-In Options

Temperature Classification	Temperature Designation	Operating Temperature °C Ambient	Burn-In 125°C (Hr)
Commercial	Null Q	0 to 70 0 to 70	None 168 ± 8
Extended	T L	-40 to +85 40 to +85	None 168 ± 8
Automotive	A	-40 to +110	None

NOTE:

Other Burn-In options are also available, but not standard.

PIN DESCRIPTIONS

Diagrams are for pin reference only. Package sizes are not to scale.

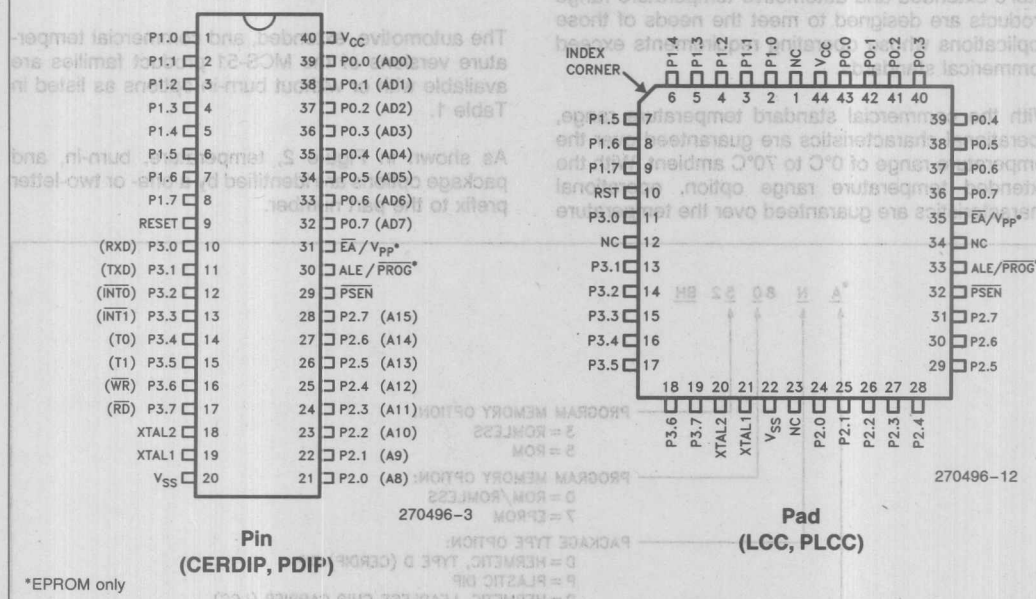


Figure 3. Pin Connections

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s, and can source and sink 8 LS TTL inputs.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL} , on the data sheet) because of the internal pullups.

In addition, P1.0 and P1.1 serve the functions of the following special features of the MCS®-51 Family:

Port Pin	Alternate Function
P1.0	T2 (Timer/Counter 2 External Input)
P1.1	T2EX (Timer/Counter 2 Capture/Reload Trigger)

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} , on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} , on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS®-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

PSEN: Program Store Enable is the Read strobe to External Program Memory.

When the device is executing code from external Program Memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to External Data Memory.

EA: External Access enable. EA must be strapped to V_{SS} in order to enable the device to fetch code from External Program Memory locations 0000H to 1FFFH. Note, however, that if either of the Lock Bits are programmed, EA will be internally latched on reset.

EA should be strapped to V_{CC} for internal program executions.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 4. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Applications Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 5. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

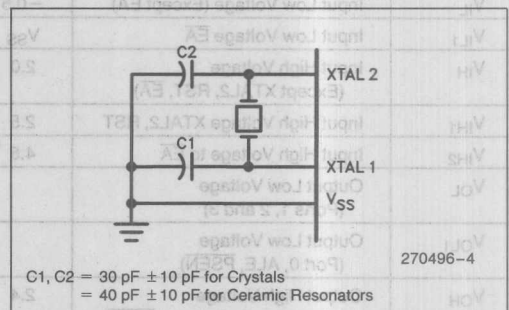


Figure 4. Oscillator Connections

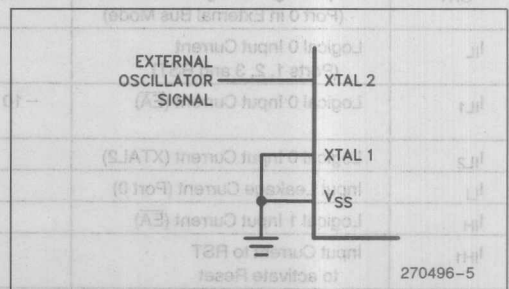


Figure 5. External Clock Drive Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias -40°C to $+110^{\circ}\text{C}$
Storage Temperature -65°C to $+150^{\circ}\text{C}$
Voltage on $\overline{\text{EA}}$ Pin
to V_{SS} -0.5V to $+13.0\text{V}$
Voltage on Any Other Pin to V_{SS} -0.5V to $+7\text{V}$
Power Dissipation 1.5W
(based on PACKAGE heat transfer limitations, not device power consumption)

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

Typical Thermal Resistance Junction to Ambient (θ_{JA})

Package	θ_{JA}
Ceramic	36°C/W
PLCC	45°C/W

ADVANCE INFORMATION. Contact Intel for Design-In Information.

D.C. CHARACTERISTICS ($T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{\text{CC}} = 5\text{V} \pm 10\%$; $V_{\text{SS}} = 0\text{V}$)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{\text{EA}}$)	-0.5	0.8	V	
V_{IL1}	Input Low Voltage $\overline{\text{EA}}$	V_{SS}	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST, $\overline{\text{EA}}$)	2.0	$V_{\text{CC}} + 0.5$	V	
V_{IH1}	Input High Voltage XTAL2, RST	2.5	$V_{\text{CC}} + 0.5$	V	XTAL1 = V_{SS}
V_{IH2}	Input High Voltage to $\overline{\text{EA}}$	4.5	5.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2 and 3)		0.45	V	$I_{\text{OL}} = 1.6\text{ mA}$ (Note 1)
V_{OL1}	Output Low Voltage (Port 0, ALE, PSEN)		0.45	V	$I_{\text{OL}} = 3.2\text{ mA}$ (Note 1)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE and PSEN)	2.4		V	$I_{\text{OH}} = -80\text{ }\mu\text{A}$
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	$I_{\text{OH}} = -400\text{ }\mu\text{A}$
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3 and RST)		-500	μA	$V_{\text{IN}} = 0.45\text{ V}$
I_{IL1}	Logical 0 Input Current ($\overline{\text{EA}}$)	-10	500	mA μA	$V_{\text{IN}} = V_{\text{SS}}$
I_{IL2}	Logical 0 Input Current (XTAL2)		-3.2	mA	$V_{\text{IN}} = 0.45\text{ V}$ XTAL1 = V_{SS}
I_{LI}	Input Leakage Current (Port 0)		± 10	μA	$0.45 < V_{\text{IN}} < V_{\text{CC}}$
I_{IH}	Logical 1 Input Current ($\overline{\text{EA}}$)		1	mA	$4.5\text{ V} < V_{\text{IN}} < 5.5\text{ V}$
I_{IH1}	Input Current to RST to activate Reset		500	μA	$V_{\text{IN}} < (V_{\text{CC}} - 1.5\text{ V})$
I_{CC}	Power Supply Current		175	mA	All Outputs Disconnected
C_{IO}	Pin Capacitance		10	pF	Test freq = 1 MHz

NOTES:

1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL} s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading $> 100\text{ pF}$), the noise pulse on the ALE pin may exceed 0.8V . In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a "T" (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A: Address.

C: Clock.

D: Input data.

H: Logic level HIGH.

I: Instruction (program memory contents).

L: Logic level LOW, or ALE.

P: PSEN.

Q: Output data.

R: RD signal.

T: Time.

V: Valid.

W: WR signal.

X: No longer a valid logic level.

Z: Float.

For example,

TAVLL = Time from Address Valid to ALE Low.

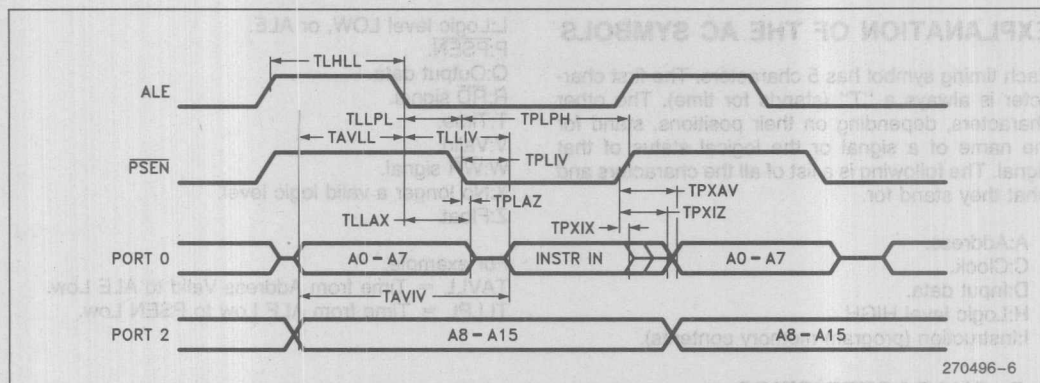
TLLPL = Time from ALE Low to PSEN Low.

A.C. CHARACTERISTICS ($T_A = -40^\circ\text{C}$ to $+110^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$); Load Capacitance for Port 0, ALE and PSEN = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

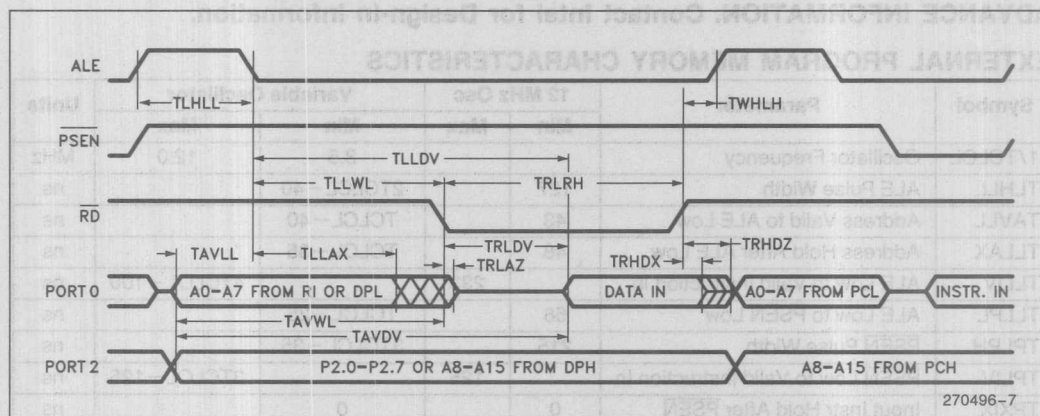
ADVANCE INFORMATION. Contact Intel for Design-In Information.

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

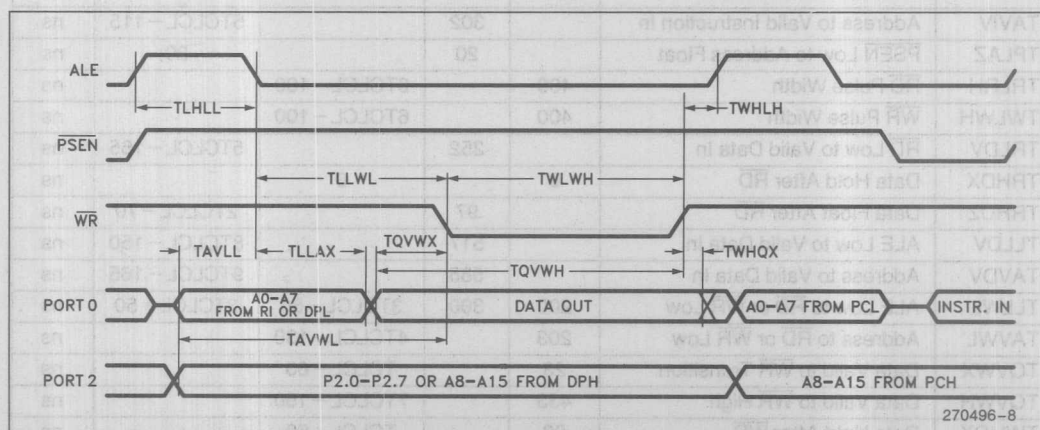
Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL - 40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL - 40		ns
TLLAX	Address Hold After ALE Low	48		TCLCL - 35		ns
TLLIV	ALE Low to Valid Instruction In		233		4TCLCL - 100	ns
TLLPL	ALE Low to PSEN Low	58		TCLCL - 25		ns
TPLPH	PSEN Pulse Width	215		3TCLCL - 35		ns
TPLIV	PSEN Low to Valid Instruction In		125		3TCLCL - 125	ns
TPXIX	Input Instr Hold After PSEN	0		0		ns
TPXIZ	Input Instr Float After PSEN		63		TCLCL - 20	ns
TPXAV	PSEN to Address Valid	75		TCLCL - 8		ns
TAVIV	Address to Valid Instruction In		302		5TCLCL - 115	ns
TPLAZ	PSEN Low to Address Float		20		20	ns
TRLRH	RD Pulse Width	400		6TCLCL - 100		ns
TWLWH	WR Pulse Width	400		6TCLCL - 100		ns
TRLDV	RD Low to Valid Data In		252		5TCLCL - 165	ns
TRHDX	Data Hold After RD	0		0		ns
TRHDZ	Data Float After RD		97		2TCLCL - 70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL - 150	ns
TAVDV	Address to Valid Data In		585		9TCLCL - 165	ns
TLLWL	ALE Low to RD or WR Low	200	300	3TCLCL - 50	3TCLCL + 50	ns
TAVWL	Address to RD or WR Low	203		4TCLCL - 130		ns
TQVWX	Data Valid to WR Transition	23		TCLCL - 60		ns
TQVWH	Data Valid to WR High	433		7TCLCL - 150		ns
TWHQX	Data Held After WR	33		TCLCL - 50		ns
TRLAZ	RD Low to Address Float		0		0	ns
TWHLH	RD or WR High to ALE High	43	123	TCLCL - 40	TCLCL + 40	ns



External Program Memory Read Cycle



External Data Memory Read Cycle



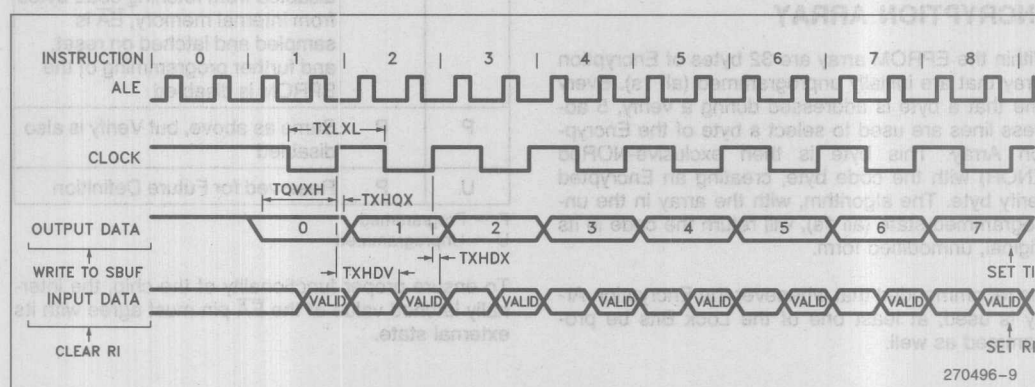
External Data Memory Write Cycle

SERIAL PORT TIMING — SHIFT REGISTER MODE

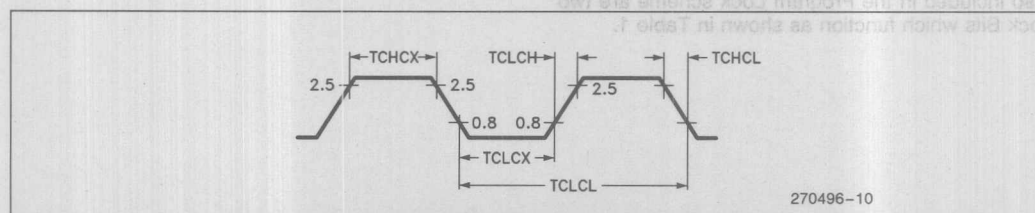
TEST CONDITIONS

$T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$; Load Capacitance = 80 pF

Symbol	Parameter	12MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL – 133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL – 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700	10TCLCL – 133		ns



Shift Register Mode Timing Waveforms



External Clock Drive Waveforms

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

PROGRAM MEMORY LOCK

The two-level Program Lock system consists of 2 Lock bits and a 32-byte Encryption Array which are used to protect the program memory against software piracy. The following description applies to the 8752BH. The same options are also available on the 8052BH, mask-programmed at the factory.

ENCRYPTION ARRAY

Within the EPROM array are 32 bytes of Encryption Array that are initially unprogrammed (all 1s). Every time that a byte is addressed during a verify, 5 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NORed (XNOR) with the code byte, creating an Encrypted Verify byte. The algorithm, with the array in the unprogrammed state (all 1s), will return the code in its original, unmodified form.

It is recommended that whenever the Encryption Array is used, at least one of the Lock Bits be programmed as well.

LOCK BITS

Also included in the Program Lock scheme are two Lock Bits which function as shown in Table 1.

AC TESTING INPUT/OUTPUT WAVEFORMS

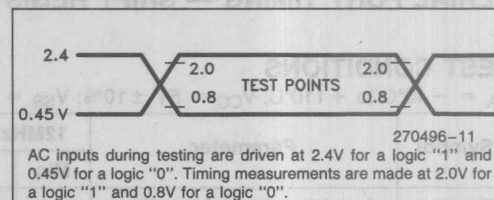


Table 1. Lock Bits and their Features

Lock Bits		Logic Enabled
LB1	LB2	
U	U	Minimum Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array)
P	U	MOV C instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled
P	P	Same as above, but Verify is also disabled
U	P	Reserved for Future Definition

P = Programmed
U = Unprogrammed

To ensure proper functionality of the chip, the internally latched value of the EA pin must agree with its external state.

8752BH

SINGLE-CHIP 8-BIT MICROCOMPUTER

WITH 8K BYTES OF EPROM PROGRAM MEMORY

AUTOMOTIVE

- Program Memory Lock
- 256 Bytes Data RAM
- Quick Pulse Programming™ Algorithm
- 12.75V Programming Voltage
- Boolean Processor
- 32 Programmable I/O Lines
- Three 16-Bit Timer/Counters
- 6 Interrupt Sources
- Programmable Serial Channel
- Separate Transmit/Receive Baud Rate Capability
- 64K External Program Memory Space
- 64K External Data Memory Space

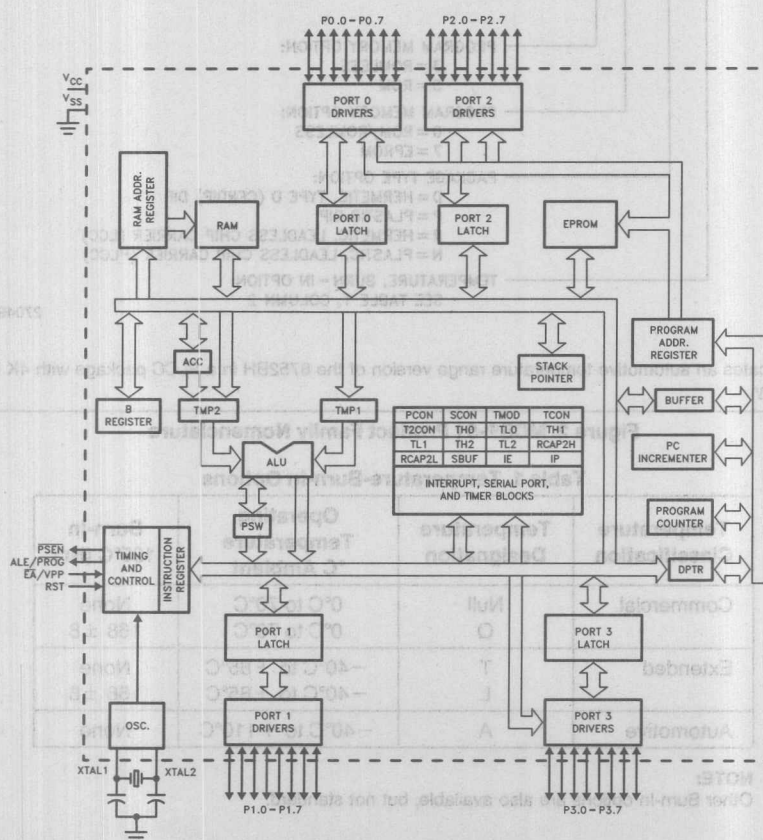


Figure 1. Block Diagram

270498-1

8752BH PRODUCT OPTIONS

Intel's extended and automotive temperature range products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to 70°C ambient. With the extended temperature range option, operational characteristics are guaranteed over the temperature

range of -40°C to +85°C ambient. For the automotive temperature range option, operational characteristics are guaranteed over the temperature range of -40°C to +125°C ambient.

The automotive, extended, and commercial temperature versions of the MCS-51 product families are available with or without burn-in options as listed in Table 1.

As shown in Figure 2, temperature, burn-in, and package options are identified by a one- or two-letter prefix to the part number.

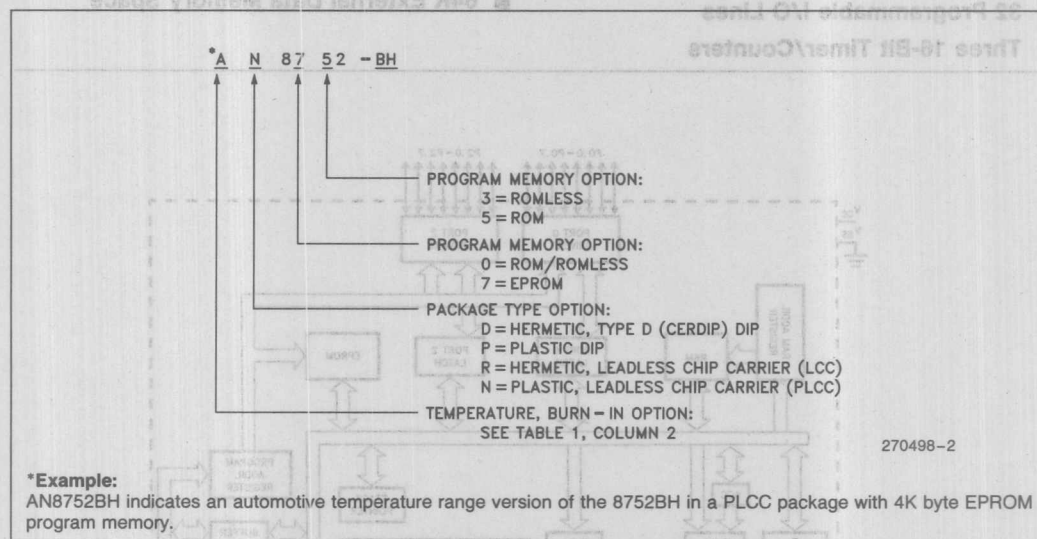


Figure 2. MCS-51 Product Family Nomenclature

Table 1. Temperature-Burn-In Options

Temperature Classification	Temperature Designation	Operating Temperature °C Ambient	Burn-In 125°C (Hr)
Commercial	Null Q	0°C to 70°C 0°C to 70°C	None 168 ± 8
Extended	T L	-40°C to +85°C -40°C to +85°C	None 168 ± 8
Automotive	A	-40°C to +110°C	None

NOTE:

Other Burn-In options are also available, but not standard.

Diagrams are for pin reference only. Package sizes are not to scale.

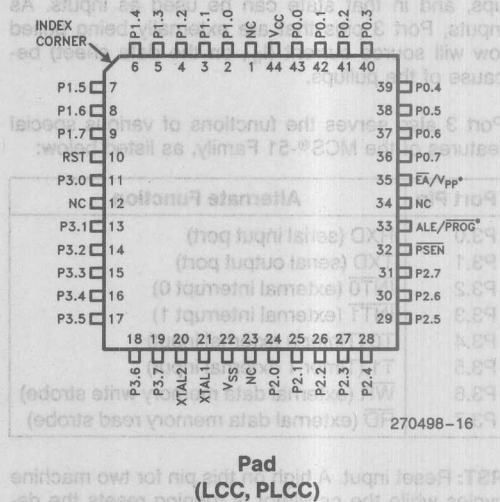
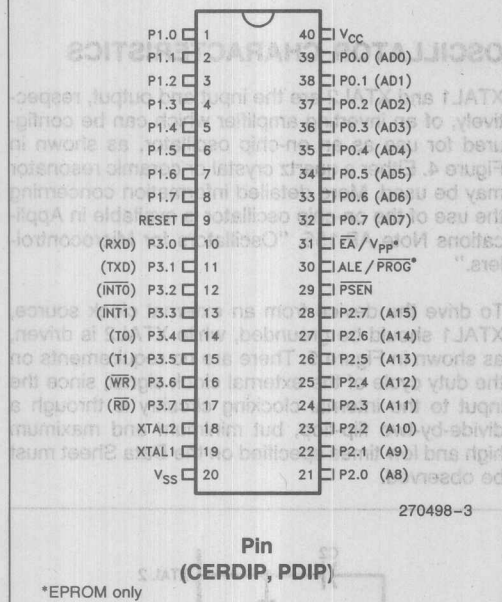


Figure 3. Pin Connections

VCC: Supply voltage.

VSS: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s, and can source and sink 8 LS TTL inputs.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL} , on the data sheet) because of the internal pullups.

In addition, P1.0 and P1.1 serve the functions of the following special features of the MCS[®]-51 Family:

Port Pin	Alternate Function
P1.0	T2 (Timer/Counter 2 External Input)
P1.1	T2EX (Timer/Counter 2 Capture/Reload Trigger)

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} , on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS[®]-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	\overline{WR} (external data memory write strobe)
P3.7	\overline{RD} (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

PSEN: Program Store Enable is the Read strobe to External Program Memory.

When the device is executing code from external Program Memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to External Data Memory.

\overline{EA} : External Access enable. \overline{EA} must be strapped to V_{SS} in order to enable the device to fetch code from External Program Memory locations 0000H to 1FFFFH. Note, however, that if either of the Lock Bits are programmed, \overline{EA} will be internally latched on reset.

\overline{EA} should be strapped to V_{CC} for internal program executions.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 4. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Applications Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 5. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

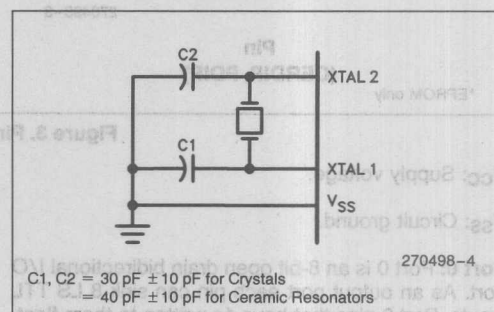


Figure 4. Oscillator Connections

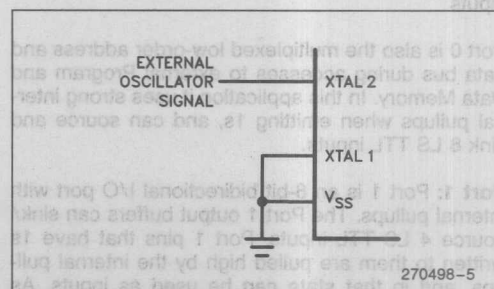


Figure 5. External Clock Drive Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature	
Under Bias	-40°C to +110°C
Storage Temperature	-65°C to +150°C
Voltage on \overline{EA} Pin	
to V_{SS}	-0.5V to +13.0V
Voltage on Any Other Pin to V_{SS}	-0.5V to +7V
Power Dissipation	1.5W
(based on PACKAGE heat transfer limitations, not device power consumption)	

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

Typical Thermal Resistance Junction to Ambient (θ_{ja})

Package	θ_{ja}
Ceramic	36°C/W
PLCC	45°C/W

ADVANCE INFORMATION—CONTACT INTEL FOR DESIGN-IN INFORMATION

D.C. CHARACTERISTICS ($T_A = -40^\circ\text{C}$ to $+110^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except \overline{EA})	-0.5	0.8	V	
V_{IL1}	Input Low Voltage \overline{EA}	V_{SS}	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST, \overline{EA})	2.0	$V_{CC} + 0.5$	V	
V_{IH1}	Input High Voltage XTAL2, RST	2.5	$V_{CC} + 0.5$	V	XTAL1 = V_{SS}
V_{IH2}	Input High Voltage to \overline{EA}	4.5	5.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2 and 3)		0.45	V	$I_{OL} = 1.6 \text{ mA}$ (Note 1)
V_{OL1}	Output Low Voltage (Port 0, ALE, \overline{PSEN})		0.45	V	$I_{OL} = 3.2 \text{ mA}$ (Note 1)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE and \overline{PSEN})	2.4		V	$I_{OH} = -80 \mu\text{A}$
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	$I_{OH} = -400 \mu\text{A}$
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3 and RST)		-500	μA	$V_{IN} = 0.45 \text{ V}$
I_{IL1}	Logical 0 Input Current (\overline{EA})	-10		mA	$V_{IN} = V_{SS}$
			500	μA	
I_{IL2}	Logical 0 Input Current (XTAL2)		-3.2	mA	$V_{IN} = 0.45 \text{ V}$ XTAL1 = V_{SS}
I_{LI}	Input Leakage Current (Port 0)		± 10	μA	$0.45 < V_{IN} < V_{CC}$
I_{IH}	Logical 1 Input Current (\overline{EA})		1	mA	$4.5 \text{ V} < V_{IN} < 5.5 \text{ V}$
I_{IH1}	Input Current to RST to activate Reset		500	μA	$V_{IN} < (V_{CC} - 1.5 \text{ V})$
I_{CC}	Power Supply Current		175	mA	All Outputs Disconnected
C_{IO}	Pin Capacitance		10	pF	Test freq = 1MHz

NOTES:

1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL} s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading $> 100 \text{ pF}$), the noise pulse on the ALE pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a "T" (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A: Address.

C: Clock.

D: Input data.

H: Logic level HIGH.

I: Instruction (program memory contents).

L: Logic level LOW, or ALE.

P: PSEN.

Q: Output data.

R: RD signal.

T: Time.

V: Valid.

W: WR signal.

X: No longer a valid logic level.

Z: Float.

For example,

TAVLL = Time from Address Valid to ALE Low.

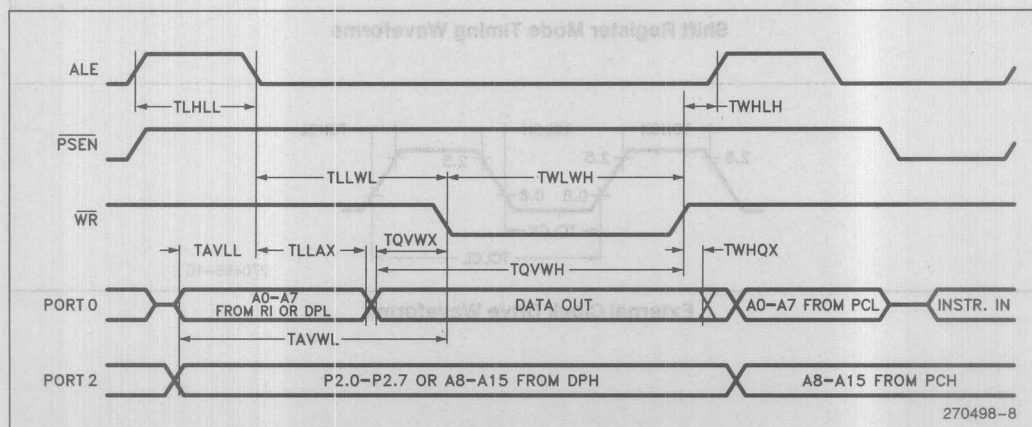
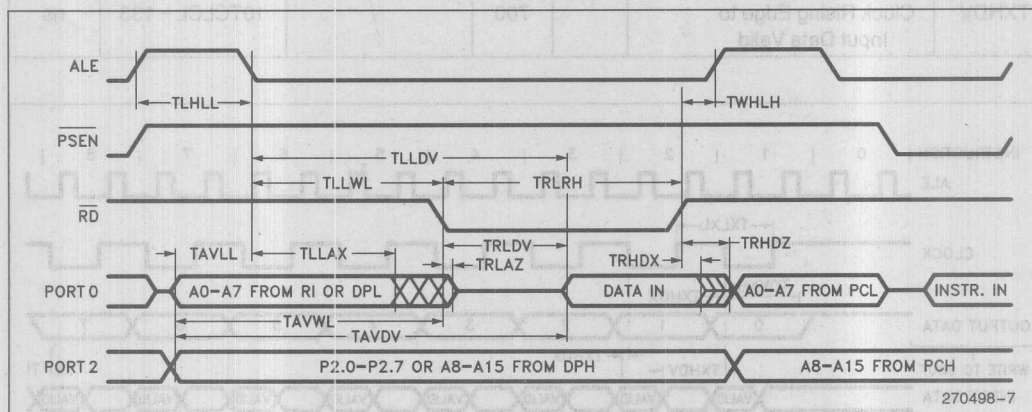
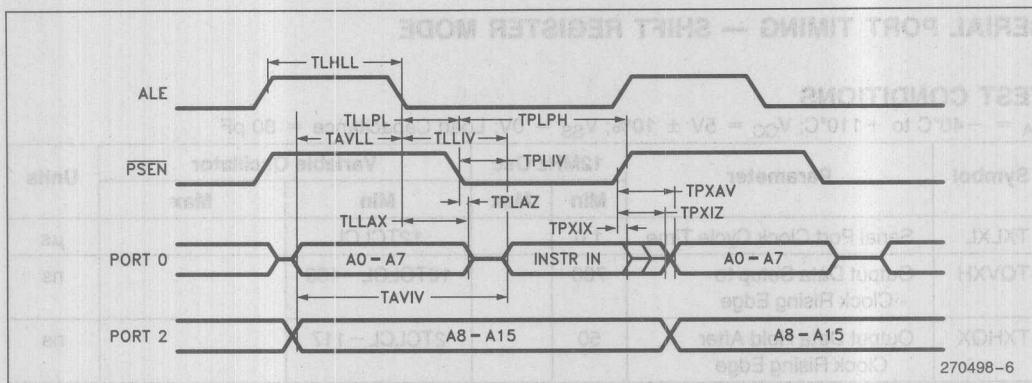
TLLPL = Time from ALE Low to PSEN Low.

A.C. CHARACTERISTICS ($T_A = -40^\circ\text{C}$ to $+110^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$); Load Capacitance for Port 0, ALE and PSEN = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

ADVANCE INFORMATION—CONTACT INTEL FOR DESIGN-IN INFORMATION

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL - 40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL - 40		ns
TLLAX	Address Hold After ALE Low	48		TCLCL - 35		ns
TLLIV	ALE Low to Valid Instruction In		233	4TCLCL - 100		ns
TLLPL	ALE Low to PSEN Low	58		TCLCL - 25		ns
TPLP	PSEN Pulse Width	215		3TCLCL - 35		ns
TPLIV	PSEN Low to Valid Instruction In		125	3TCLCL - 125		ns
TPXIX	Input Instr Hold After PSEN	0		0		ns
TPXIZ	Input Instr Float After PSEN		63	TCLCL - 20		ns
TPXAV	PSEN to Address Valid	75		TCLCL - 8		ns
TAVIV	Address to Valid Instruction In		302	5TCLCL - 115		ns
TPLAZ	PSEN Low to Address Float		20	20		ns
TRLRH	RD Pulse Width	400		6TCLCL - 100		ns
TWLWH	WR Pulse Width	400		6TCLCL - 100		ns
TRLDV	RD Low to Valid Data In		252	5TCLCL - 165		ns
TRHDX	Data Hold After RD	0		0		ns
TRHDZ	Data Float After RD		97	2TCLCL - 70		ns
TLLDV	ALE Low to Valid Data In		517	8TCLCL - 150		ns
TAVDV	Address to Valid Data In		585	9TCLCL - 165		ns
TLLWL	ALE Low to RD or WR Low	200	300	3TCLCL - 50	3TCLCL + 50	ns
TAVWL	Address to RD or WR Low	203		4TCLCL - 130		ns
TQVWX	Data Valid to WR Transition	23		TCLCL - 60		ns
TQVWH	Data Valid to WR High	433		7TCLCL - 150		ns
TWHQX	Data Held After WR	33		TCLCL - 50		ns
TRLAZ	RD Low to Address Float		0	0		ns
TWHLH	RD or WR High to ALE High	43	123	TCLCL - 40	TCLCL + 40	ns

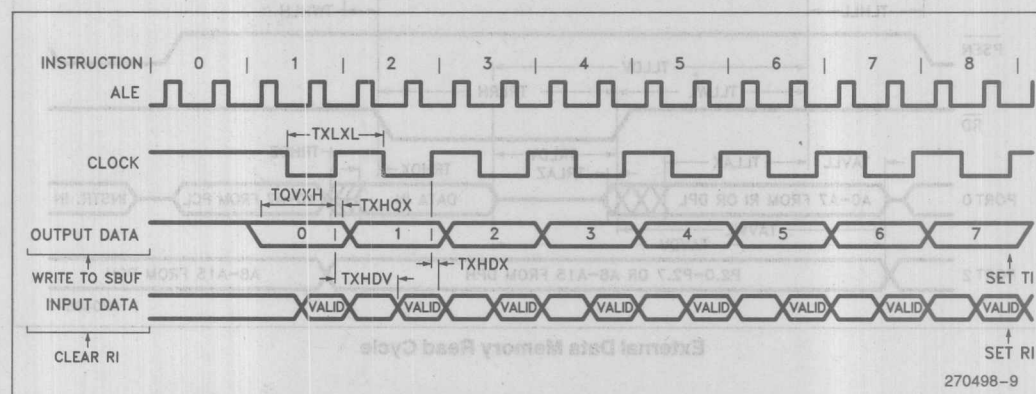


SERIAL PORT TIMING — SHIFT REGISTER MODE

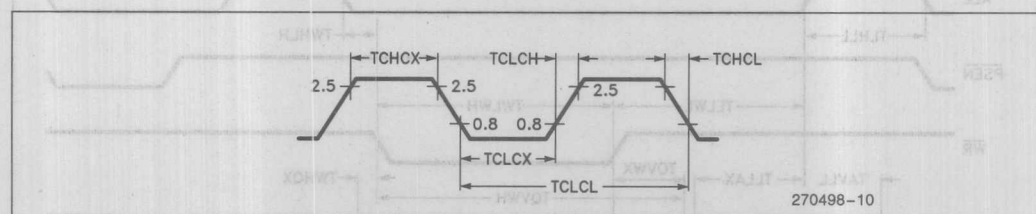
TEST CONDITIONS

$T_A = -40^{\circ}\text{C}$ to $+110^{\circ}\text{C}$; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$; Load Capacitance = 80 pF

Symbol	Parameter	12MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL - 133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL - 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL - 133	ns



Shift Register Mode Timing Waveforms



External Clock Drive Waveforms

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

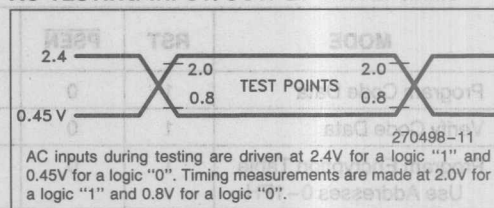
EPROM CHARACTERISTICS

Table 1 shows the logic levels for programming the Program Memory, the Encryption Table, and the Lock Bits and for reading the signature bytes.

Programming the EPROM

To be programmed, the 8752BH must be running with a 4 MHz to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal registers.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0–P2.4 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 and 3 pins, and RST,

AC TESTING INPUT/OUTPUT WAVEFORMS



PSEN, and \overline{EA}/V_{PP} should be held at the "Program" levels indicated in Table 2. ALE/ \overline{PROG} is pulsed low to program the code byte into the addressed EPROM location. The setup is shown in Figure 6.

Normally \overline{EA}/V_{PP} is held at a logic high until just before ALE/ \overline{PROG} is to be pulsed. Then \overline{EA}/V_{PP} is raised to V_{PP} , ALE/ \overline{PROG} is pulsed low, and then \overline{EA}/V_{PP} is returned to a valid high voltage. The voltage on the \overline{EA}/V_{PP} pin must be at the valid \overline{EA}/V_{PP} high level before a verify is attempted. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

Note that the \overline{EA}/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches.

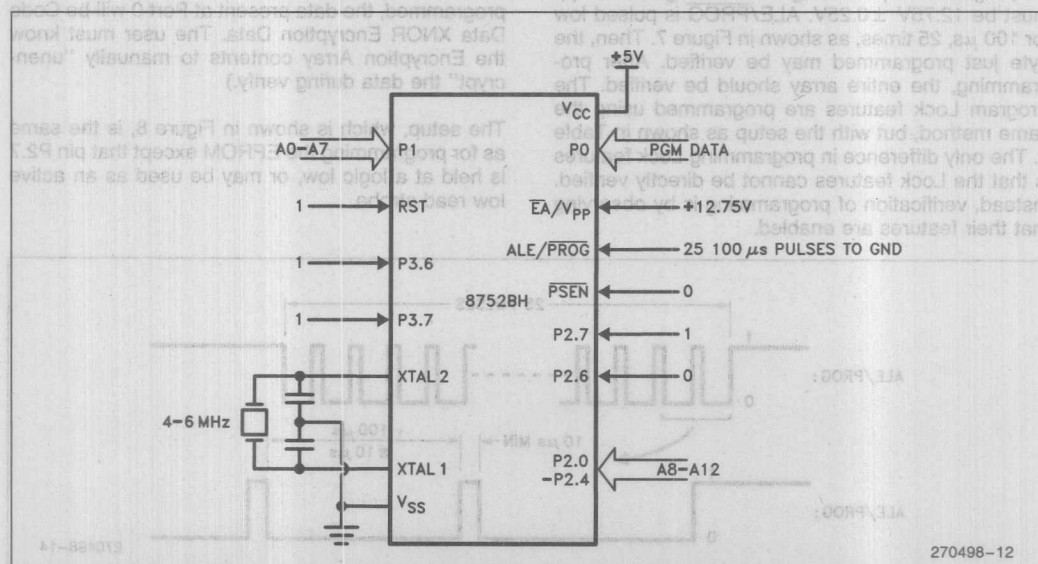


Figure 6. Programming the EPROM

Table 2. EPROM Programming Modes

MODE	RST	PSEN	ALE/ PROG	EA/ V _{PP}	P2.7	P2.6	P3.6	P3.7
Program Code Data	1	0	0*	V _{PP}	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Program Encryption Table Use Addresses 0-1FH	1	0	0*	V _{PP}	1	0	0	1
Program Lock x = 1	1	0	0*	V _{PP}	1	1	1	1
Bits (LBx) x = 2	1	0	0*	V _{PP}	1	1	0	0
Read Signature	1	0	1	1	0	0	0	0

NOTES:

"1" = Valid high for that pin

"0" = Valid low for that pin

"V_{PP}" = +12.75V ±0.25V

*ALE/PROG is pulsed low for 100 μs for programming. (Quick-Pulse Programming™)

**QUICK-PULSE PROGRAMMING™
ALGORITHM**

The 8752BH can be programmed using the Quick-Pulse Programming™ Algorithm for microcontrollers. The features of the new programming method are a lower V_{PP} (12.75V as compared to 21V) and a shorter programming pulse. It is possible to program the entire 8 Kbytes of EPROM memory in less than 25 seconds with this algorithm!

To program the part using the new algorithm, V_{PP} must be 12.75V ±0.25V. ALE/PROG is pulsed low for 100 μs, 25 times, as shown in Figure 7. Then, the byte just programmed may be verified. After programming, the entire array should be verified. The Program Lock features are programmed using the same method, but with the setup as shown in Table 2. The only difference in programming Lock features is that the Lock features cannot be directly verified. Instead, verification of programming is by observing that their features are enabled.

PROGRAM VERIFICATION

If the Lock Bits have not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0-P2.4. The other pins should be held at the "Verify" levels indicated in Table 2. The contents of the addressed location will come out on Port 0. External pullups are required on Port 0 for this operation. (If the Encryption Array in the EPROM has been programmed, the data present at Port 0 will be Code Data XNOR Encryption Data. The user must know the Encryption Array contents to manually "unencrypt" the data during verify.)

The setup, which is shown in Figure 8, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active low read strobe.

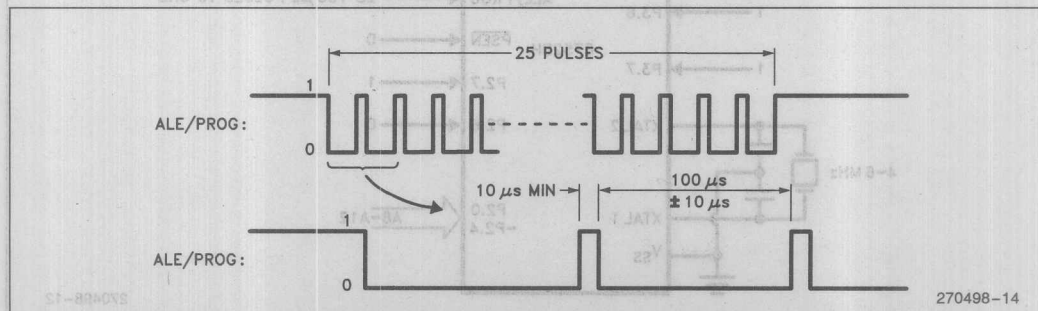


Figure 7. PROG Waveforms

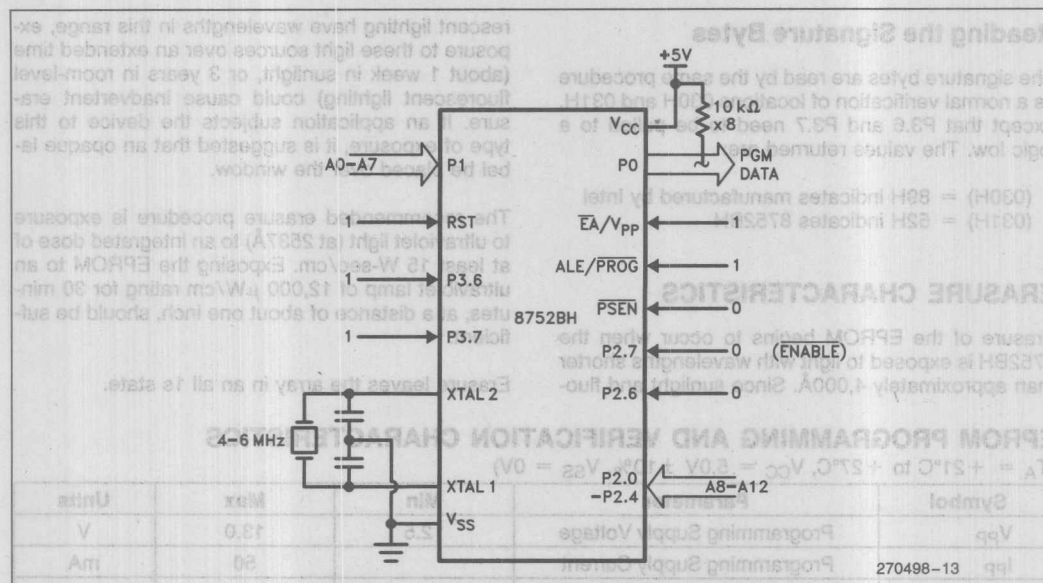


Figure 8. Verifying the EPROM

PROGRAM MEMORY LOCK

The two-level Program Lock system consists of 2 Lock bits and a 32-byte Encryption Array which are used to protect the program memory against software piracy.

ENCRYPTION ARRAY

Within the EPROM array are 32 bytes of Encryption Array that are initially unprogrammed (all 1s). Every time that a byte is addressed during a verify, 5 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NORed (XNOR) with the code byte, creating an Encrypted Verify byte. The algorithm, with the array in the unprogrammed state (all 1s), will return the code in its original, unmodified form.

It is recommended that whenever the Encryption Array is used, at least one of the Lock Bits be programmed as well.

LOCK BITS

Also included in the EPROM Program Lock scheme are two Lock Bits which function as shown in Table 3.

Table 3. Lock Bits and Their Features

Lock Bits		Logic Enabled
LB1	LB2	
U	U	Minimum Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array)
P	U	MOVX instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled
P	P	Same as above, but Verify is also disabled
U	P	Reserved for Future Definition

P = Programmed
U = Unprogrammed

Erasing the EPROM also erases the Encryption Array and the Lock Bits, returning the part to full unlocked functionality.

To ensure proper functionality of the chip, the internally latched value of the EA pin must agree with its external state.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values returned are:

(030H) = 89H indicates manufactured by Intel
(031H) = 52H indicates 8752BH

ERASURE CHARACTERISTICS

Erasure of the EPROM begins to occur when the 8752BH is exposed to light with wavelengths shorter than approximately 4,000Å. Since sunlight and fluo-

rescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

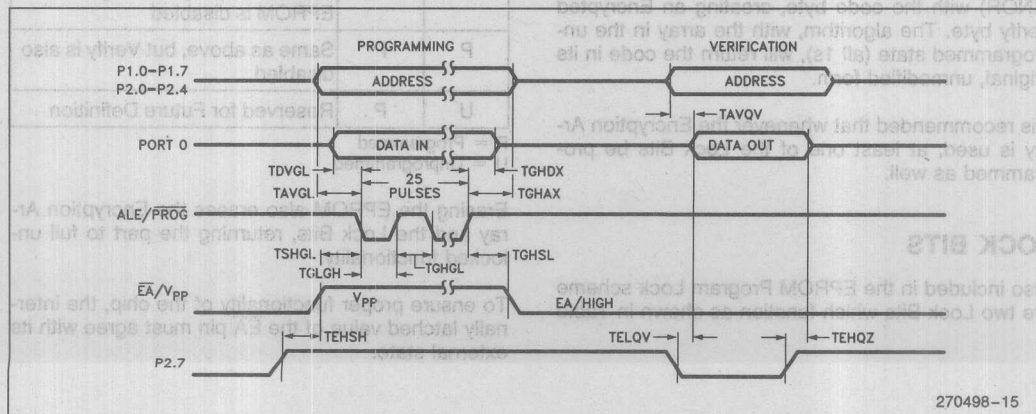
The recommended erasure procedure is exposure to ultraviolet light (at 2537Å) to an integrated dose of at least 15 W-sec/cm. Exposing the EPROM to an ultraviolet lamp of 12,000 μ W/cm rating for 30 minutes, at a distance of about one inch, should be sufficient.

Erasure leaves the array in an all 1s state.

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

($T_A = +21^\circ\text{C}$ to $+27^\circ\text{C}$, $V_{CC} = 5.0\text{V} \pm 10\%$, $V_{SS} = 0\text{V}$)

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	12.5	13.0	V
I_{PP}	Programming Supply Current		50	mA
$1/\text{TCLCL}$	Oscillator Frequency		6	MHz
TAVGL	Address Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHAX	Address Hold after $\overline{\text{PROG}}$	48TCLCL		
TDVGL	Data Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHDX	Data Hold after $\overline{\text{PROG}}$	48TCLCL		
TEHSH	P2.7 ($\overline{\text{ENABLE}}$ High to V_{PP})	48TCLCL		
TSHGL	V_{PP} Setup to $\overline{\text{PROG}}$ Low	10		μ s
TGHSL	V_{PP} Hold after $\overline{\text{PROG}}$	10		μ s
TGLGH	$\overline{\text{PROG}}$ Width	90	110	μ s
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{\text{ENABLE}}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float after $\overline{\text{ENABLE}}$	0	48TCLCL	
TGHGL	$\overline{\text{PROG}}$ High to $\overline{\text{PROG}}$ Low	10		μ s



270498-15

EPROM Programming and Verification Waveforms

MCS®-51

80C31BH/80C51BH/87C51

CHMOS SINGLE-CHIP 8-BIT MICROCONTROLLER

AUTOMOTIVE

- Extended Automotive Temperature Range (–40°C to +125°C Ambient)
- High Performance CHMOS Process
- Power Control Modes
- 4K byte On-Chip ROM/EPROM
- 128 x 8-bit RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- 5 Interrupt Sources
- Quick-Pulse EPROM Programming
- 2-Level Program Memory Lock EPROM

- Boolean Processor
- Programmable Serial Channel
- TTL- and CMOS-Compatible Logic Levels
- 64K External Program Memory Space
- 64K External Data Memory Space
- IDLE and POWER DOWN Modes
- ONCE™ Mode Facilitates System Testing (EPROM only)
- Available in LCC, PLCC and DIP Packages

(See Packaging Specification, Order #231369)

The MCS®-51 CHMOS products are fabricated on Intel's CHMOS III (ROM) and CHMOS II-E (EPROM) processes and are functionally compatible with the standard MCS-51 HMOS and EPROM products. This technology combines the high speed and density characteristics of HMOS with the low power attributes of CHMOS. This combination expands the effectiveness of the powerful MCS-51 architecture and instruction set.

Like the MCS-51 HMOS versions, the MCS-51 CHMOS products have the following features: 4K bytes of EPROM/ROM (87C51/80C51BH respectively); 128 bytes of RAM; 32 I/O lines; two 16-bit timer/counters; a five-source two-level interrupt structure; a full duplex serial port; and on-chip oscillator and clock circuitry. In addition, the MCS-51 CHMOS products exhibit low operating power, along with two software selectable modes of reduced activity for further power reduction—Idle and Power Down.

The Idle mode freezes the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power Down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

The 87C51 is the EPROM version of the 80C51BH. It contains 4K bytes of on-chip program memory that can be electrically programmed, and can be erased by exposure to ultraviolet light. The 87C51 EPROM array uses a modified Quick-Pulse Programming™ algorithm, by which the entire 4K byte array can be programmed in about 12 seconds.

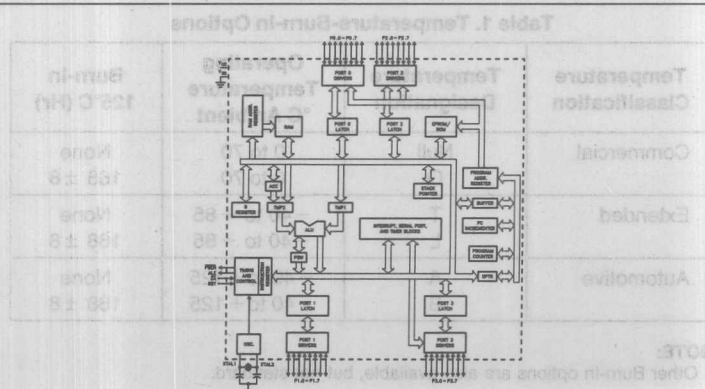


Figure 1. MCS®-51 Architectural Block Diagram

87C51/80C51BH/80C31BH PRODUCT OPTIONS

Intel's extended and automotive temperature range products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to 70°C ambient. With the extended temperature range option, operational characteristics are guaranteed over the temperature range of -40°C to +85°C ambient. For the automotive temperature range option, operational characteristics are guaranteed over the temperature range of -40°C to +125°C ambient.

teristics are guaranteed over the temperature range of -40°C to +125°C ambient.

The automotive, extended, and commercial temperature versions of the MCS-51 product families are available with or without burn-in options as listed in Table 1.

As shown in Figure 2, temperature, burn-in, and package options are identified by a one- or two-letter prefix to the part number.

Operating frequency options are noted with a number suffix: The standard operating frequency of 12 MHz is designated without a suffix; the 16 MHz operating frequency version is designated by the -1 suffix.

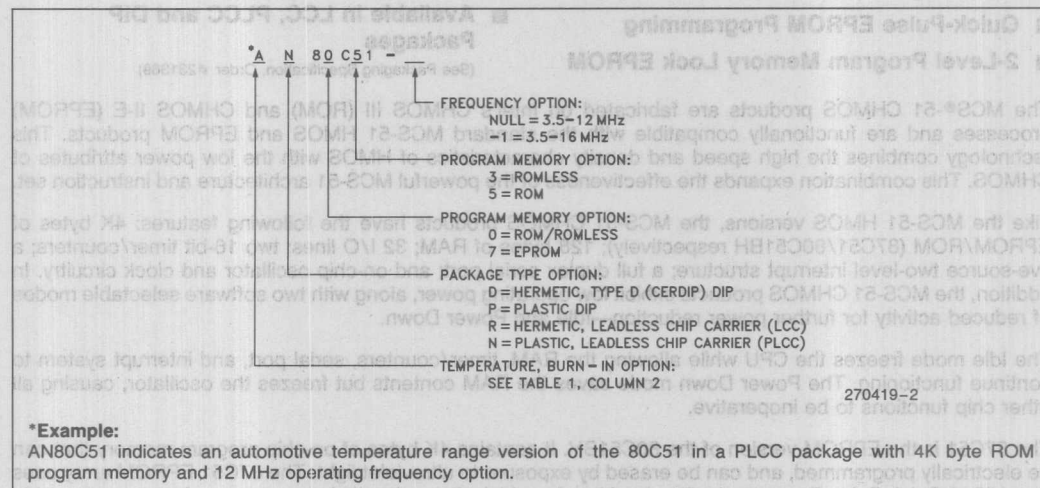


Figure 2. MCS®-51 Product Family Nomenclature

Table 1. Temperature-Burn-In Options

Temperature Classification	Temperature Designation	Operating Temperature °C Ambient	Burn-In 125°C (Hr)
Commercial	Null	0 to 70	None
	Q	0 to 70	168 ± 8
Extended	T	-40 to +85	None
	L	-40 to +85	168 ± 8
Automotive	A	-40 to +125	None
	B	-40 to +125	168 ± 8

NOTE:

Other Burn-In options are also available, but not standard.

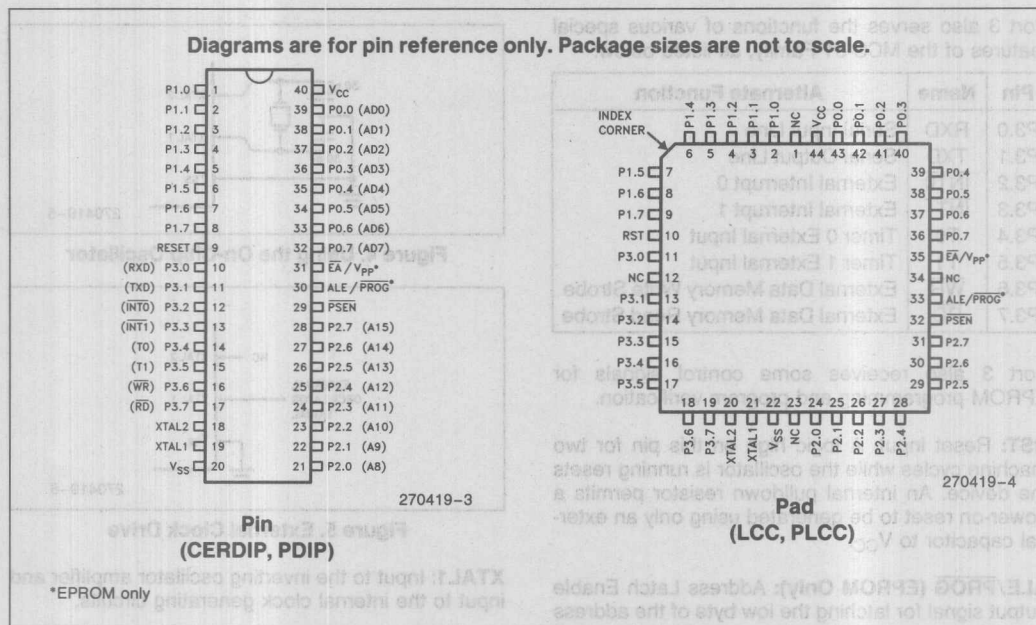


Figure 3. Pin Connections

PIN DESCRIPTION

V_{CC}: Supply voltage during normal, Idle, and Power Down operations.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external memory. In this application it uses strong internal pullups when emitting 1s.

Port 0 also receives the code bytes during EPROM programming, and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL} , on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during EPROM programming and program verification.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} , on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program memory and during accesses to external Data Memory that use 16-bit address (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s.

During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives some control signals and the high-order address bits during EPROM programming and program verification.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} , on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Pin	Name	Alternate Function
P3.0	RXD	Serial Input Line
P3.1	TXD	Serial Output Line
P3.2	INT0	External Interrupt 0
P3.3	INT1	External Interrupt 1
P3.4	T0	Timer 0 External Input
P3.5	T1	Timer 1 External Input
P3.6	WR	External Data Memory Write Strobe
P3.7	RD	External Data Memory Read Strobe

Port 3 also receives some control signals for EPROM programming and program verification.

RST: Reset input. A logic high on this pin for two machine cycles while the oscillator is running resets the device. An internal pulldown resistor permits a power-on reset to be generated using only an external capacitor to V_{CC} .

ALE/PROG (EPROM Only): Address Latch Enable output signal for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during EPROM programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

PSEN: Program Store Enable is the Read strobe to External Program Memory. When the 87C51/80C51BH is executing from Internal Program Memory, PSEN is inactive (high). When the device is executing code from External Program Memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to External Data Memory.

EA/Vpp: External Access enable. EA must be strapped to V_{SS} in order to enable the 87C51/80C51BH to fetch code from External Program Memory locations 0000H to 0FFFH. [Note, however, that if either of the Lock Bits is programmed, the logic level at EA is internally latched during reset.] (EPROM only.)

EA must be strapped to V_{CC} for internal program execution.

Vpp (EPROM Only): This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.

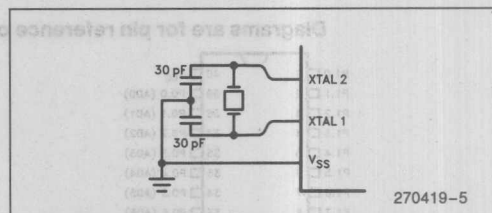


Figure 4. Using the On-Chip Oscillator

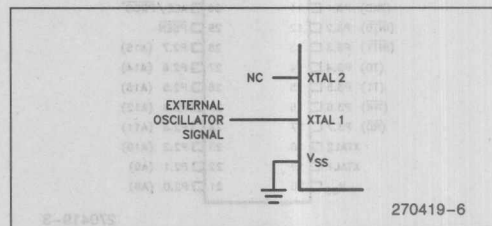


Figure 5. External Clock Drive

XTAL1: Input to the inverting oscillator amplifier and input to the internal clock generating circuits.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 4.

To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected, as shown in Figure 5. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

IDLE MODE

In Idle Mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the Special Functions Registers remain unchanged during this mode. The Idle Mode can be terminated by any enabled interrupt or by a hardware reset.

It should be noted that when Idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to

Table 2. Status of the external pins during Idle and Power Down

Mode	Program Memory	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

NOTE:

For more detailed information on the reduced power modes refer to current Embedded Controller Handbook, and Application Note AP-252, "Designing with the 80C51BH."

internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

POWER DOWN MODE

In the Power Down mode the oscillator is stopped, and the instruction that invokes Power Down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power Down mode is terminated.

The only exit from Power Down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

DESIGN CONSIDERATIONS (EPROM Only)

Exposure to light when the device is in operation may cause logic errors. For this reason, it is suggested that an opaque label be placed over the window when the die is exposed to ambient light.

PROGRAM MEMORY LOCK (EPROM Only)

The 87C51 contains two program memory lock schemes: Encrypted Verify and Lock Bits.

Encrypted Verify: The 87C51 implements a 32-byte EPROM array that can be programmed by the customer, and which can then be used to encrypt the program code bytes during EPROM verification. The EPROM verification procedure is performed as usual, except that each code byte comes out logically X-NORed with one of the 32 key bytes. The key bytes are gone through in sequence. Therefore, to read the ROM code, one has to know the 32 key bytes in their proper sequence.

Lock Bits: Also on the chip are two Lock Bits which can be left unprogrammed (U) or can be programmed (P) to obtain the following additional features:

Bit 1	Bit 2	Additional Features
U	U	none
P	U	<ul style="list-style-type: none"> Externally fetched code can not access internal Program Memory. Further programming disabled.
U	P	(Reserved for Future definition.)
P	P	<ul style="list-style-type: none"> Externally fetched code can not access internal Program Memory. Further programming disabled. Program verification is disabled.

When Lock Bit 1 is programmed, the logic level at the EA pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of EA be in agreement with the current logic level at that pin in order for the device to function properly.

ONCETM MODE (EPROM Only)

The ONCETM ("on-circuit emulation") mode facilitates testing and debugging of systems using the 87C51 without the 87C51 having to be removed from the circuit. The ONCE mode is invoked by:

1. Pull ALE low while the device is in reset and PSEN is high;
2. Hold ALE low as RST is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins and ALE and PSEN are weakly pulled high. The oscillator circuit remains active. While the 87C51 is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored when a normal reset is applied.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	-40°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on \overline{EA}/V_{PP} Pin to V_{SS}	0V to +13.0V
Voltage on Any Other Pin to V_{SS}	-0.5V to +6.5V
Power Dissipation	1.5W
(Based on package heat transfer limitations, not device power consumption).	
Typical Junction Temperature (T_J)	+135°C
(Based upon ambient temperature at +125°C)	
Typical Thermal Resistance Junction-to-Ambient (θ_{JA}):	
CERDIP	36°C/W
LCC	38°C/W
PDIP	75°C/W
PLCC	46°C/W

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS: ($T_A = -40^\circ\text{C}$ to $+125^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$)

Symbol	Parameter	Min	Typ(1)	Max (87C51/80C51BH)	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except \overline{EA})	-0.5		$0.2 V_{CC} - 0.25$	V	
V_{IL1}	Input Low Voltage to \overline{EA}	0		$0.2 V_{CC} - 0.45$	V	
V_{IH}	Input High Voltage (Except XTAL1, RST)	$0.2 V_{CC} + 1.0$		$V_{CC} + 0.5$	V	
V_{IH1}	Input High Voltage (XTAL1, RST)	$0.7 V_{CC} + 0.1$		$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage (Ports 1, 2, 3)			0.45	V	$I_{OL} = 1.6 \text{ mA}^{(2)}$
V_{OL1}	Output Low Voltage (Port 0, ALE, PSEN)			0.45	V	$I_{OL} = 3.2 \text{ mA}^{(2)}$
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, PSEN)	2.4			V	$I_{OH} = -60 \mu\text{A}$
		$0.9 V_{CC}$			V	$I_{OH} = -10 \mu\text{A}$
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4			V	$I_{OH} = -800 \mu\text{A}$
		$0.9 V_{CC}$			V	$I_{OH} = -80 \mu\text{A}^{(3)}$
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3)			-75	μA	$V_{IN} = 0.45 \text{ V}$
I_{TL}	Logical 1-to-0 transition current (Ports 1, 2, 3)			-750	μA	(4)
I_{LI}	Input Leakage Current (Port 0)			± 10	μA	$V_{IN} = V_{IL}$ or V_{IH}
I_{CC}	Power Supply Current: Active Mode @ 12 MHz (5) Idle Mode @ 12 MHz (5) Power Down Mode		11.5 1.3 3	35/20 6/2.5 150/75 (7)	mA mA μA	(6) $V_{CC} = 2.2\text{V to } 5.5\text{V}$
RRST	Internal Reset Pulldown Resistor	50		300	K Ω	
CIO	Pin Capacitance			10	pF	

*Versions of the 80C51BH and 80C31BH are available with $V_{CC} = 5V \pm 20\%$ upon request.

NOTES:

- "Typicals" are based on a limited number of samples taken from early manufacturing lots and are not guaranteed. The values listed are at room temp, 5V.
- Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL} s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading $> 100\text{pF}$), the noise pulse on the ALE pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
- Capacitive loading on Ports 0 and 2 may cause the V_{OH} on ALE and PSEN to momentarily fall below the $0.9 V_{CC}$ specification when the address bits are stabilizing.

NOTES: (Continued)

4. Pins of Ports 1, 2, and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. ICCMAX at other frequencies is given by:

$$\text{Active Mode: } ICCMAX = 0.94 \times \text{FREQ} + 23.71$$

$$\text{Idle Mode: } ICCMAX = 0.14 \times \text{FREQ} + 3.81$$

where FREQ is the external oscillator frequency in MHz. ICCMAX is given in mA. See Figure 6.

6. See Figures 7 through 10 for I_{CC} test conditions.

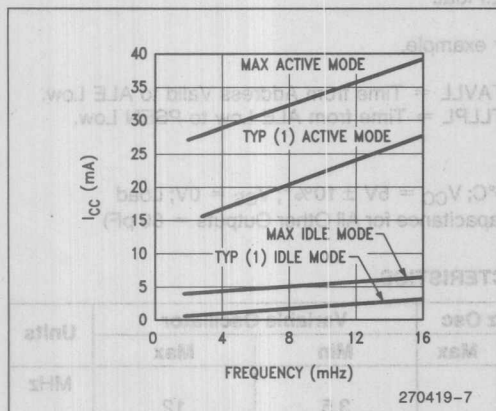


Figure 6. I_{CC} vs. FREQ. Valid only within frequency specifications of the device under test.

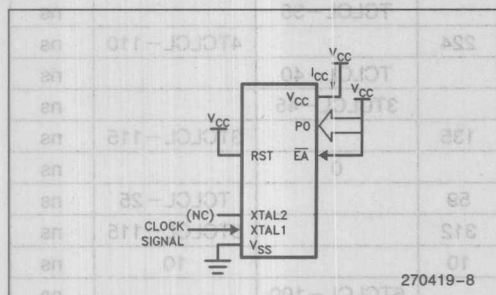


Figure 7. I_{CC} Test Condition, Active Mode. All other pins are disconnected.

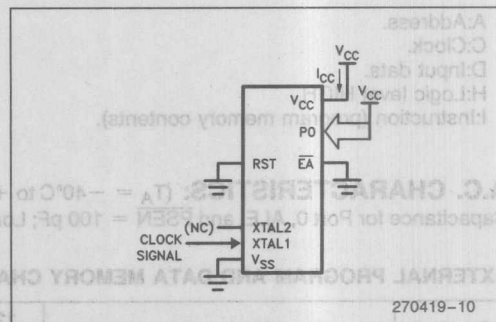


Figure 8. I_{CC} Test Condition, Idle Mode. All other pins are disconnected.

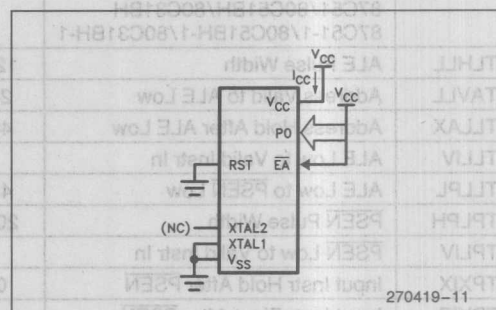


Figure 9. I_{CC} Test Condition, Power Down Mode. All other pins are disconnected.

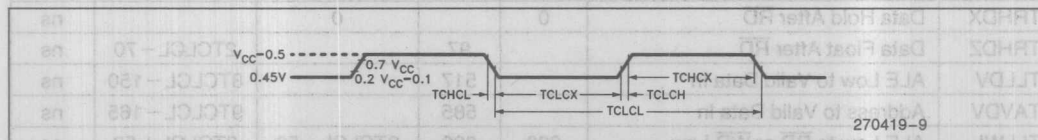


Figure 10. Clock Signal Waveform for I_{CC} tests in Active and Idle Modes. $TCLCH = TCHCL = 5$ ns.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A: Address.
C: Clock.
D: Input data.
H: Logic level HIGH.
I: Instruction (program memory contents).

L: Logic level LOW, or ALE.

P: PSEN.

Q: Output data.

R: RD signal.

T: Time.

V: Valid.

W: WR signal.

X: No longer a valid logic level.

Z: Float.

For example,

TAVLL = Time from Address Valid to ALE Low.

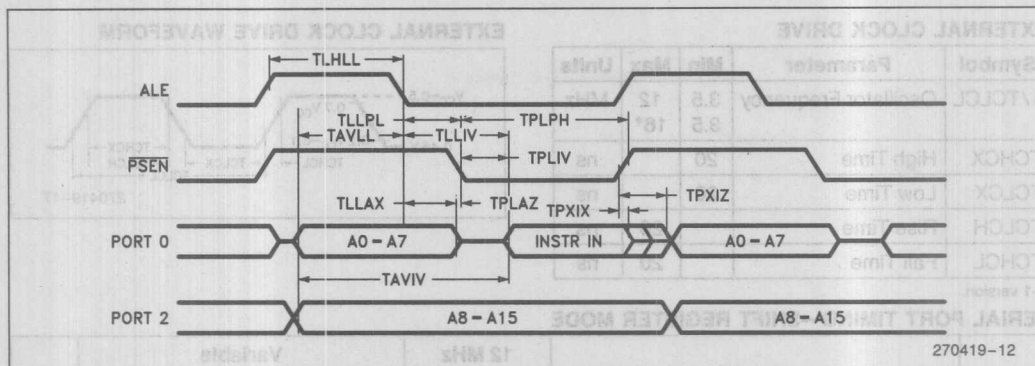
TLLPL = Time from ALE Low to PSEN Low.

A.C. CHARACTERISTICS: ($T_A = -40^\circ\text{C}$ to $+125^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$; Load Capacitance for Port 0, ALE, and PSEN = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

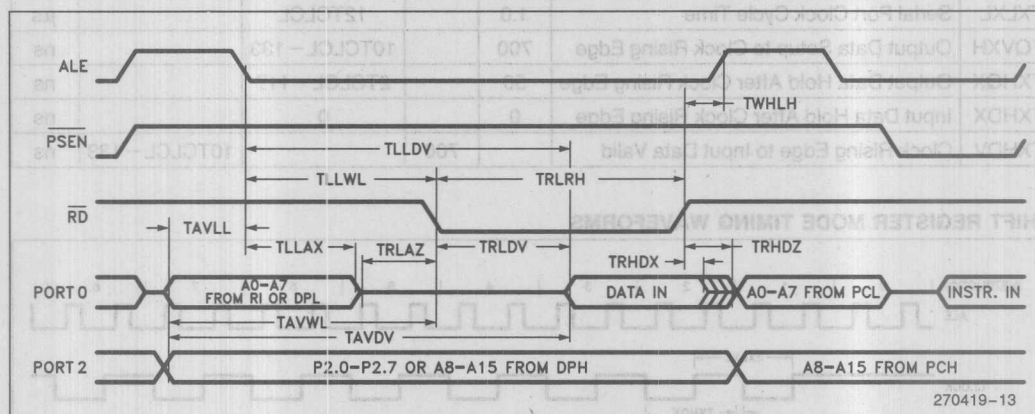
EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency 87C51/80C51BH/80C31BH 87C51-1/80C51BH-1/80C31BH-1			3.5 3.5	12 16	MHz
TLHLL	ALE Pulse Width	127		2TCLCL - 40		ns
TAVLL	Address Valid to ALE Low	28		TCLCL - 55		ns
TLLAX	Address Hold After ALE Low	48		TCLCL - 35		ns
TLLIV	ALE Low to Valid Instr In		224		4TCLCL - 110	ns
TLLPL	ALE Low to PSEN Low	43		TCLCL - 40		ns
TPLPH	PSEN Pulse Width	205		3TCLCL - 45		ns
TPLIV	PSEN Low to Valid Instr In		135		3TCLCL - 115	ns
TPXIX	Input Instr Hold After PSEN	0		0		ns
TPXIZ	Input Instr Float After PSEN		59		TCLCL - 25	ns
TAVIV	Address to Valid Instr In		312		5TCLCL - 115	ns
TPLAZ	PSEN Low to Address Float		10		10	ns
TRLRH	RD Pulse Width	400		6TCLCL - 100		ns
TWLWH	WR Pulse Width	400		6TCLCL - 100		ns
TRLDV	RD Low to Valid Data In		252		5TCLCL - 175	ns
TRHDX	Data Hold After RD	0		0		ns
TRHDZ	Data Float After RD		97		2TCLCL - 70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL - 150	ns
TAVDV	Address to Valid Data In		585		9TCLCL - 165	ns
TLLWL	ALE Low to RD or WR Low	200	300	3TCLCL - 50	3TCLCL + 50	ns
TAVWL	Address to RD or WR Low	203		4TCLCL - 130		ns
TQVWX	Data Valid to WR Transition	23		TCLCL - 60		ns
TWHQX	Data Hold After WF	33		TCLCL - 50		ns
TRLAZ	RD Low to Address Float		0		0	ns
TWHLH	RD or WR High to ALE High	43	123	TCLCL - 40	TCLCL + 40	ns

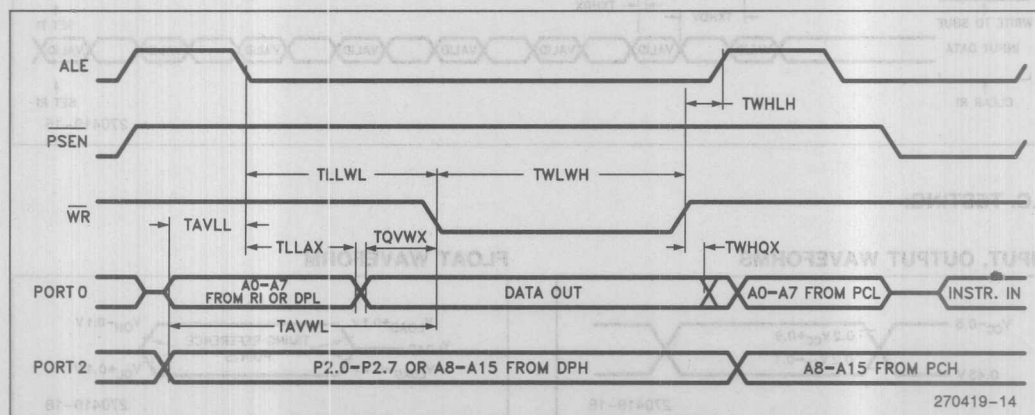
*Versions of the 80C51BH and 80C31BH are available with $V_{CC} = 5V \pm 20\%$ upon request.



External Program Memory Read Cycle



External Data Memory Read Cycle



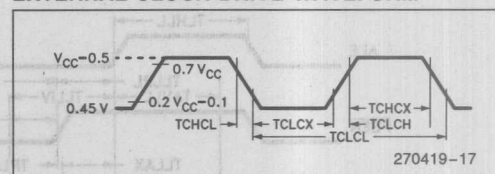
External Data Memory Write Cycle

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5 3.5	12 16*	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

* -1 version.

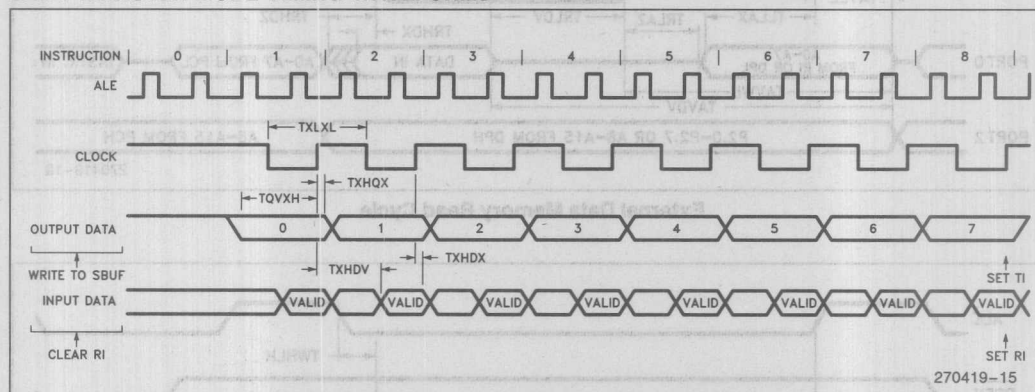
EXTERNAL CLOCK DRIVE WAVEFORM



SERIAL PORT TIMING—SHIFT REGISTER MODE

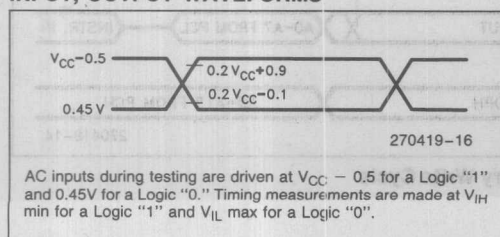
Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL - 133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL - 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL - 133	ns

SHIFT REGISTER MODE TIMING WAVEFORMS

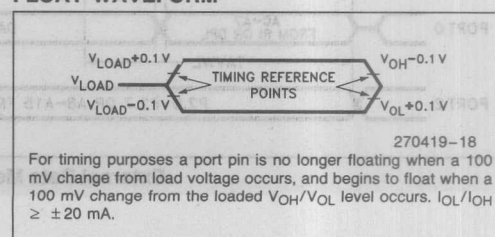


A.C. TESTING:

INPUT, OUTPUT WAVEFORMS



FLOAT WAVEFORM



EPROM CHARACTERISTICS (EPROM Only)

The 87C51 is programmed by a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (Programming Supply Voltage) and in the width and number of the ALE/PROG pulses.

The 87C51 contains two signature bytes that can be read and used by an EPROM programming system

to identify the device. The signature bytes identify the device as an 87C51 manufactured by Intel.

Table 3 shows the logic levels for reading the signature byte, and for programming the Program Memory, the Encryption Table, and the Lock Bits. The circuit configuration and waveforms for Quick-Pulse Programming™ are shown in Figures 11 and 12. Figure 13 shows the circuit configuration for normal Program Memory verification.

Table 3. EPROM Programming Modes

MODE	RST	PSEN	ALE/ PROG	EA/ V _{PP}	P2.7	P2.6	P3.7	P3.6
Read Signature	1	0	1	1	0	0	0	0
Program Code Data	1	0	0*	V _{PP}	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Pgm Encryption Table	1	0	0*	V _{PP}	1	0	1	0
Pgm Lock Bit 1	1	0	0*	V _{PP}	1	1	1	1
Pgm Lock Bit 2	1	0	0*	V _{PP}	1	1	0	0

NOTES:

"1" = Valid high for that pin

"0" = Valid low for that pin

$V_{PP} = 12.75V \pm 0.25V$

$V_{CC} = 5V \pm 10\%$ during programming and verification

*ALE/PROG receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100 $\mu S (\pm 10 \mu S)$ and high for a minimum of 10 μS .

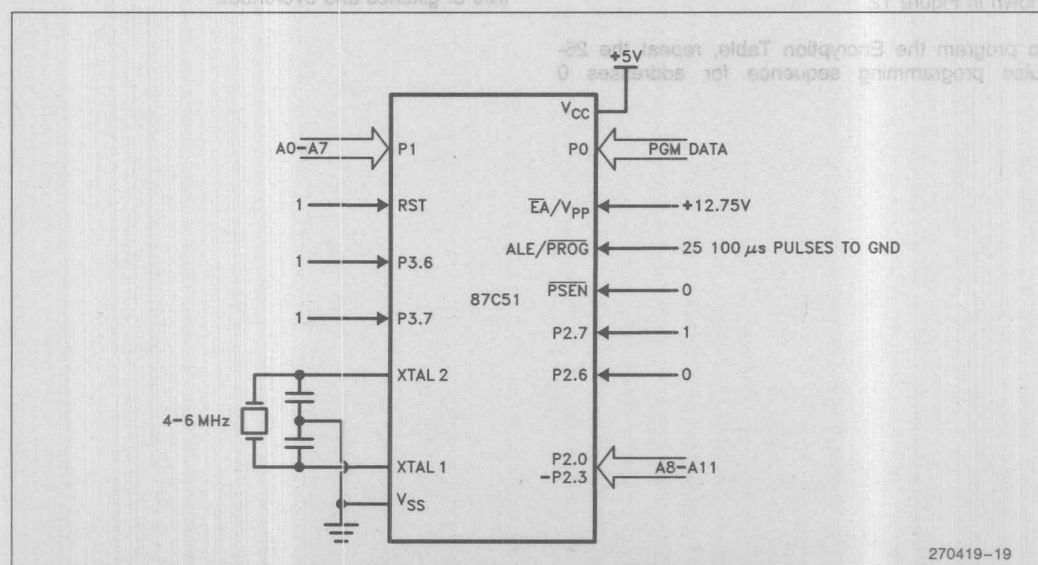


Figure 11. Programming Configuration

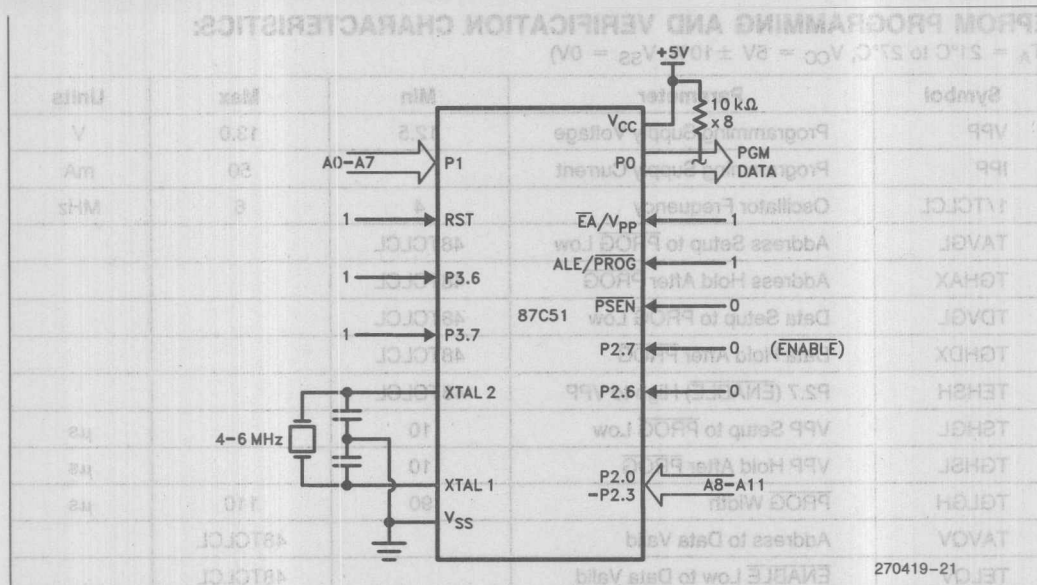


Figure 13. Program Verification

Program Verification (EPROM Only)

If Lock Bit 2 has not been programmed, the on-chip Program Memory can be read out for program verification. The address of the Program Memory location to be read is applied to Ports 1 and 2 as shown in Figure 13. The other pins are held at the "Verify Code Data" levels indicated in Table 3. The contents of the addressed location will be emitted on Port 0. External pullups are required on Port 0 for this operation. Detailed timing specifications are shown in later sections of this data sheet.

If the Encryption Table has been programmed, the data presented at Port 0 will be the Exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the Encryption Table contents in order to correctly decode the verification data. The Encryption Table itself can not be read out.

Reading the Signature Bytes (EPROM Only)

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values returned are:

- (030H) = 89H indicates manufactured by Intel
- (031H) = 57H indicates 87C51

Program/Verify Algorithms (EPROM Only)

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics (EPROM Only)

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm². Exposing the EPROM to an ultraviolet lamp of 12,000 μW/cm² rating for 30 minutes, at a distance of about 1 inch, should be sufficient.

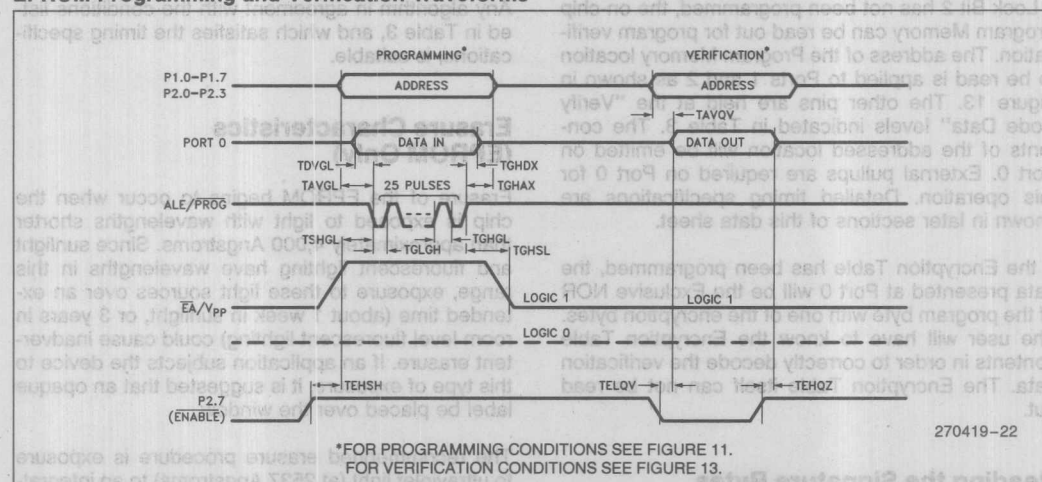
Erasure leaves the array in an all 1s state.

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS:

($T_A = 21^\circ\text{C}$ to 27°C , $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$)

Symbol	Parameter	Min	Max	Units
VPP	Programming Supply Voltage	12.5	13.0	V
IPP	Programming Supply Current		50	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHAX	Address Hold After $\overline{\text{PROG}}$	48TCLCL		
TDVGL	Data Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHDX	Data Hold After $\overline{\text{PROG}}$	48TCLCL		
TEHSH	P2.7 ($\overline{\text{ENABLE}}$) High to VPP	48TCLCL		
TSHGL	VPP Setup to $\overline{\text{PROG}}$ Low	10		μs
TGHSL	VPP Hold After $\overline{\text{PROG}}$	10		μs
TGLGH	$\overline{\text{PROG}}$ Width	90	110	μs
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{\text{ENABLE}}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float After $\overline{\text{ENABLE}}$	0	48TCLCL	
TGHGL	$\overline{\text{PROG}}$ High to $\overline{\text{PROG}}$ Low	10		μs

EPROM Programming and Verification Waveforms



80C51FA/83C51FA/87C51FA CHMOS SINGLE-CHIP 8-BIT MICROCONTROLLER WITH PROGRAMMABLE COUNTER ARRAY, UP/DOWN COUNTER, 8K BYTES USER PROGRAMMABLE EPROM/MASK ROM

- Extended Automotive Temperature Range (-40°C to $+125^{\circ}\text{C}$ Ambient)
- High Performance CHMOS EPROM
- Power Control Modes
- Three 16-Bit Timer/Counters
- Programmable Counter Array with:
 - High Speed Output,
 - Compare/Capture,
 - Pulse Width Modulator,
 - Watchdog Timer Capabilities
- Up/Down Timer/Counter
- Two Level Program Lock System
- 8K On-Chip EPROM
- 256 Bytes of On-Chip Data RAM
- Quick Pulse Programming™ Algorithm
- Boolean Processor
- 32 Programmable I/O Lines
- 7 Interrupt Sources
- Programmable Serial Channel with:
 - Framing Error Detection
 - Automatic Address Recognition
- TTL Compatible Logic Levels
- 64K External Program Memory Space
- 64K External Data Memory Space
- MCS®-51 Fully Compatible Instruction Set
- Power Saving Idle and Power Down Modes
- ONCE™ (On-Circuit Emulation) Mode

MEMORY ORGANIZATION

PROGRAM MEMORY: Up to 8K bytes of the program memory can reside in the on-chip ROM/EPROM. In addition the device can address up to 64K of program memory external to the chip.

DATA MEMORY: This microcontroller has a 256 x 8 on-chip RAM. In addition it can address up to 64K bytes of external data memory.

The Intel 80C51FA/83C51FA/87C51FA is a single-chip control oriented microcontroller which is fabricated on Intel's CHMOS III (83C51FA) ROM, and CHMOS II-E (87C51FA) EPROM technology. For the remainder of this data sheet references to the ROMless (80C51FA), ROM (83C51FA) and EPROM (87C51FA) versions will be denoted as 83C51FA. Being a member of the MCS®-51 family, the 83C51FA uses the same powerful instruction set, has the same architecture, and is pin for pin compatible with the existing MCS-51 products. The 83C51FA is an enhanced version of the 87C51. It's added features make it an even more powerful microcontroller for applications that require Pulse Width Modulation, High Speed I/O, and up/down counting capabilities such as motor control. It also has a more versatile serial channel that facilitates multi-processor communications.

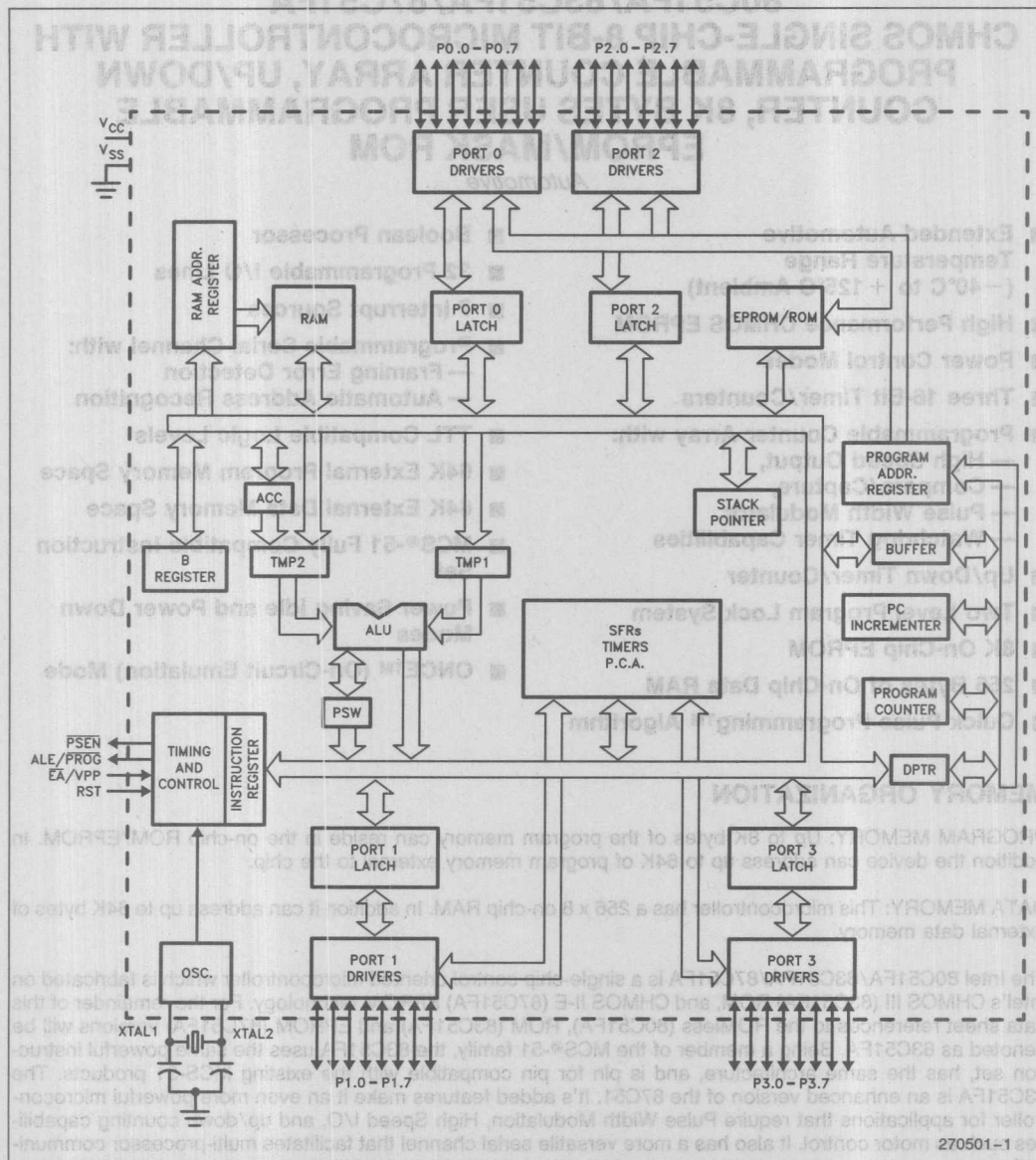


Figure 1. 83C51FA Block Diagram

80C51FA/83C51FA/87C51FA PRODUCT OPTIONS

Intel's extended and automotive temperature range products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to 70°C ambient. With the extended temperature range option, operational

characteristics are guaranteed over the temperature range of -40°C to +85°C ambient. For the automotive temperature range option, operational characteristics are guaranteed over the temperature range of -40°C to +125°C ambient. The automotive, extended, and commercial temperature versions of the MCS-51 product families are available with or without burn-in options as listed in Table 1.

As shown in Figure 2 temperature, burn-in, and package options are identified by a one- or two-letter prefix to the part number.

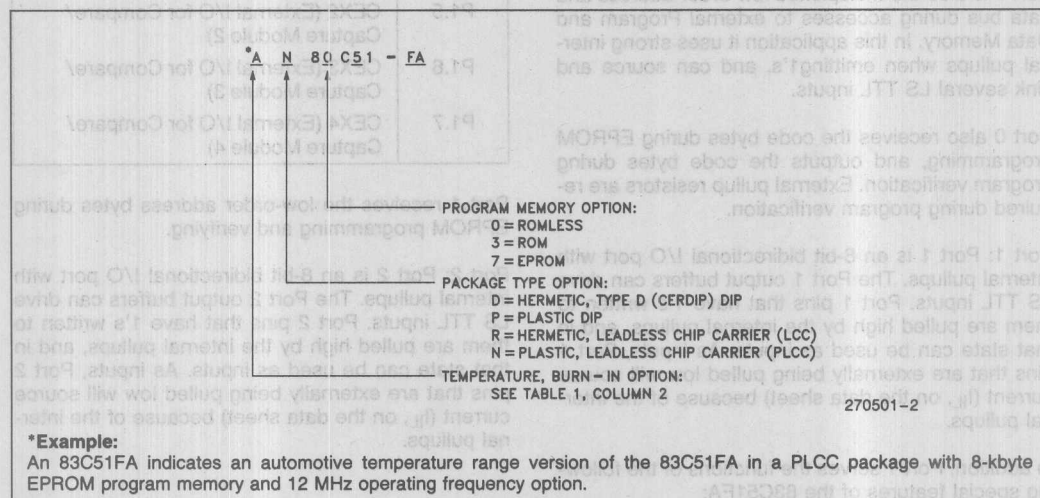


Figure 2. MCS®-51 Product Family Nomenclature

Table 1. Temperature-Burn-In Options

Temperature Classification	Temperature Designation	Operating Temperature °C Ambient	Burn-In 125°C (Hr)
Commercial	Null	0 to 70	None
	Q	0 to 70	168 ± 8
Extended	T	-40 to +85	None
	L	-40 to +85	168 ± 8
Automotive	A	-40 to +125	None
	B	-40 to +125	168 ± 8

NOTE:

Other Burn-In options are also available, but not standard.

PIN DESCRIPTIONS

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit, open drain, bidirectional I/O port. As an output port each pin can sink several LS TTL inputs. Port 0 pins that have 1's written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1's, and can source and sink several LS TTL inputs.

Port 0 also receives the code bytes during EPROM programming, and outputs the code bytes during program verification. External pullup resistors are required during program verification.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can drive LS TTL inputs. Port 1 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

In addition, Port 1 serves the functions of the following special features of the 83C51FA:

Port Pin	Alternate Function
P1.0	T2 (External Count Input to Timer/Counter 2)
P1.1	T2EX (Timer/Counter 2 Capture/Reload Trigger and Direction Control)
P1.2	ECI (External Count Input to the PCA)
P1.3	CEX0 (External I/O for Compare/Capture Module 0)
P1.4	CEX1 (External I/O for Compare/Capture Module 1)
P1.5	CEX2 (External I/O for Compare/Capture Module 2)
P1.6	CEX3 (External I/O for Compare/Capture Module 3)
P1.7	CEX4 (External I/O for Compare/Capture Module 4)

Port 1 receives the low-order address bytes during EPROM programming and verifying.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can drive LS TTL inputs. Port 2 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

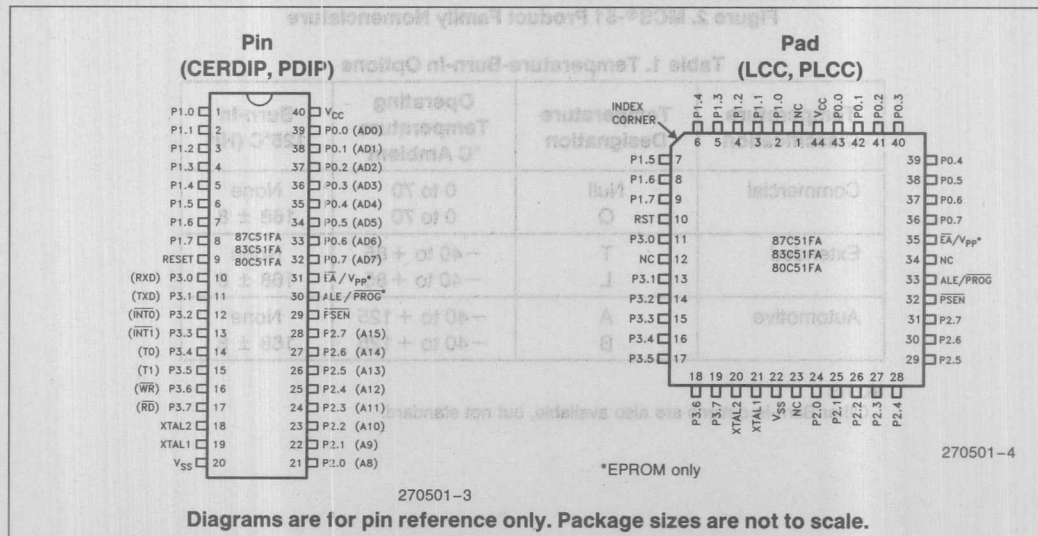


Figure 3. Pin Connections

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1's. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Some Port 2 pins receive the high-order address bits during EPROM programming and program verification.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can drive LS TTL inputs. Port 3 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. An internal pulldown resistor permits a power-on reset with only a capacitor connected to V_{CC} .

ALE/PROG: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin (ALE/PROG) is also the program pulse input during EPROM programming for the 87C51FA.

In normal operation ALE is emitted at a constant rate of $1/6$ the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

Throughout the remainder of this data sheet, ALE will refer to the signal coming out of the ALE/PROG pin, and the pin will be referred to as the ALE/PROG pin.

PSEN: Program Store Enable is the read strobe to external Program Memory.

When the 83C51FA is executing code from external Program Memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external Data Memory.

EA/Vpp: External Access enable. EA must be strapped to V_{SS} in order to enable the device to fetch code from external Program Memory locations 0000H to 1FFFFH. Note, however, that if either of the Program Lock bits are programmed, EA will be internally latched on reset.

EA should be strapped to V_{CC} for internal program executions.

This pin also receives the programming supply voltage (V_{pp}) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of a inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 4. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 floats, as shown in Figure 5. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the data sheet must be observed.

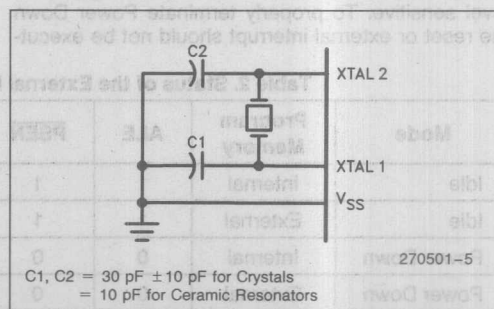


Figure 4. Oscillator Connections

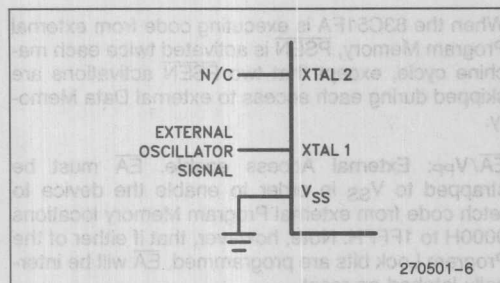


Figure 5. External Clock Drive Configuration

IDLE MODE

The user's software can invoke the Idle Mode. When the microcontroller is in this mode, power consumption is reduced. The Special Function Registers and the onboard RAM retain their values during Idle, but the processor stops executing instructions. Idle Mode will be exited if the chip is reset or if an enabled interrupt occurs. The PCA timer/counter can optionally be left running or paused during Idle Mode.

POWER DOWN MODE

To save even more power, a Power Down mode can be invoked by software. In this mode, the oscillator is stopped and the instruction that invoked Power Down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power Down mode is terminated.

On the 83C51FA either a hardware reset or an external interrupt can cause an exit from Power Down. Reset redefines all the SFRs but does not change the on-chip RAM. An external interrupt allows both the SFRs and on-chip RAM to retain their values. The interrupt must be enabled and configured as level sensitive. To properly terminate Power Down the reset or external interrupt should not be executed before V_{CC} is restored to its normal operating level, and must be held active long enough for the oscillator to restart and stabilize.

ed before V_{CC} is restored to its normal operating level, and must be held active long enough for the oscillator to restart and stabilize.

DESIGN CONSIDERATION

- Ambient light is known to affect the internal RAM contents during operation. If the 87C51FA application requires the part to be run under ambient lighting, an opaque label should be placed over the window to exclude light.
- When the Idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

ONCE™ MODE

The ONCE ("On-Circuit Emulation") Mode facilitates testing and debugging of systems using the 87C51FA without the 87C51FA having to be removed from the circuit. The ONCE Mode is invoked by:

- 1) Pull ALE low while the device is in reset and PSEN is high;
- 2) Hold ALE low as RST is deactivated.

While the device is in ONCE Mode, the Port 0 pins go into a float state, and the other port pins and ALE and PSEN are weakly pulled high. The oscillator circuit remains active. While the 83C51FA is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored when a normal reset is applied.

Table 2. Status of the External Pins during Idle and Power Down

Mode	Program Memory	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

NOTE:

For more detailed information on the reduced power modes refer to current Embedded Controller Handbook, and Application Note AP-252, "Designing with the 80C51BH."

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature	Under Bias	-40°C to +125°C
Storage Temperature		-65°C to +150°C
Voltage on EA/V _{PP} Pin to V _{SS}		0V to +13.0V*
Voltage on Any Other Pin to V _{SS}		-0.5V to +6.5V
Power Dissipation		1.5W
(Based on PACKAGE heat transfer limitations, not device power consumption)		
*EPROM only		

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

ADVANCED INFORMATION—CONTACT INTEL FOR DESIGN-IN INFORMATION

D.C. CHARACTERISTICS: (T_A = -40°C to +125°C; V_{CC} = 5V ± 10%; V_{SS} = 0V)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V _{IL}	Input Low Voltage	-0.5	0.2 V _{CC} - 0.1	V	
V _{IL1}	Input Low Voltage EA	0	0.2 V _{CC} - 0.3	V	
V _{IH}	Input High Voltage (Except XTAL2, RST, EA)	0.2 V _{CC} + 0.9	V _{CC} + 0.5	V	
V _{IH1}	Input High Voltage (XTAL, RST)	0.7 V _{CC}	V _{CC} + 0.5	V	
V _{OL}	Output Low Voltage (Ports 1, 2 and 3)		0.45	V	I _{OL} = 1.6 mA(1)
V _{OL1}	Output Low Voltage (Port 0, ALE/PROG, PSEN)		0.45	V	I _{OL} = 3.2 mA(1)
V _{OH}	Output High Voltage (Ports 1, 2 and 3 ALE/PROG and PSEN)	2.4		V	I _{OH} = -60 μA
		0.9 V _{CC}		V	I _{OH} = -10 μA(2)
V _{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	I _{OH} = -800 μA
		0.9 V _{CC}		V	I _{OH} = -80 μA(2)
I _{IL}	Logical 0 Input Current (Ports 1, 2, and 3)		-50	μA	V _{IN} = 0.45V
I _{LI}	Input leakage Current (Port 0 and EA)		± 10	μA	V _{IN} = V _{IL} or V _{IH}
I _{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, and 3)		-650	μA	V _{IN} = 2V
RRST	RST Pulldown Resistor	40	225	KΩ	
CIO	Pin Capacitance		10	pF	@1MHz, 25°C
I _{CC}	Power Supply Current: Running at 12 MHz (Figure 5) Idle Mode at 12 MHz (Figure 5) Power Down Mode		40	mA	(Note 3)
			12	mA	
			100	μA	

NOTES:

- Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1 to 0 transitions during bus operations. In applications where capacitance loading exceeds 100 pFs, the noise pulse on the ALE signal may exceed 0.8V. In these cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an Address Latch with a Schmitt Trigger Strobe input.
- Capacitive loading on Ports 0 and 2 cause the V_{OH} on ALE and PSEN to drop below the 0.9 V_{CC} specification when the address lines are stabilizing.
- See Figures 6-9 for test conditions.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A: Address
C: Clock
D: Input Data
H: Logic level HIGH
I: Instruction (program memory contents)

L: Logic level LOW, or ALE
P: PSEN
Q: Output Data
R: \overline{RD} signal
T: Time
V: Valid
W: \overline{WR} signal
X: No longer a valid logic level
Z: Float

For example,

TAVLL = Time from Address Valid to ALE Low
TLLPL = Time from ALE Low to PSEN Low

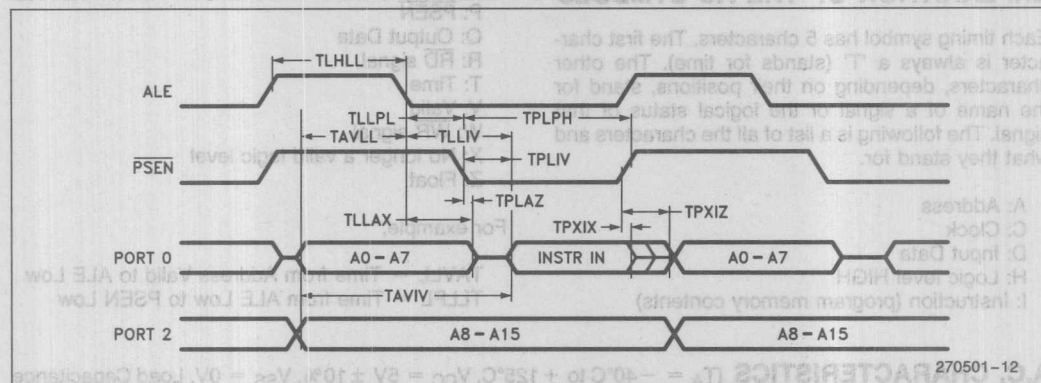
A.C. CHARACTERISTICS ($T_A = -40^\circ\text{C}$ to $+125^\circ\text{C}$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$, Load Capacitance for Port 0, ALE/PROG and PSEN = 100 pF, Load Capacitance for All Other Outputs = 80 pF)

ADVANCED INFORMATION—CONTACT INTEL FOR DESIGN-IN INFORMATION

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

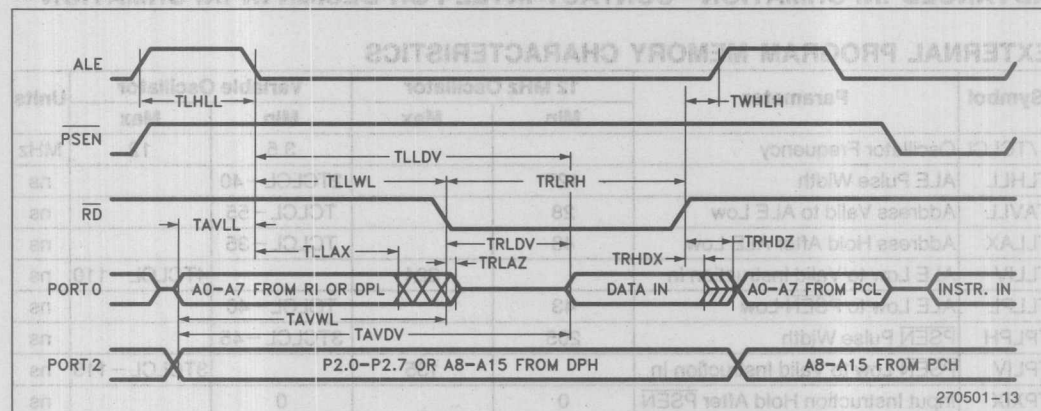
Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12	MHz
TLHLL	ALE Pulse Width	127		2TCLCL - 40		ns
TAVLL	Address Valid to ALE Low	28		TCLCL - 55		ns
TLLAX	Address Hold After ALE Low	48		TCLCL - 35		ns
TLLIV	ALE Low to Valid Instruction In		224		4TCLCL - 110	ns
TLLPL	ALE Low to PSEN Low	43		TCLCL - 40		ns
TPLPH	PSEN Pulse Width	205		3TCLCL - 45		ns
TPLIV	PSEN Low to Valid Instruction In		135		3TCLCL - 115	ns
TPXIX	Input Instruction Hold After PSEN	0		0		ns
TPXIZ	Input Instruction Float After PSEN		59		TCLCL - 25	ns
TAVIV	Address to Valid Instruction In		302		5TCLCL - 115	ns
TPLAZ	PSEN Low to Address Float		10		10	ns
TRLRH	\overline{RD} Pulse Width	400		6TCLCL - 100		ns
TWLWH	\overline{WR} Pulse Width	400		6TCLCL - 100		ns
TRLDV	\overline{RD} Low to Valid Data In		242		5TCLCL - 175	ns
TRHDX	Data Hold After \overline{RD}	-10		-10		ns
TRHDZ	Data Float After \overline{RD}		97		2TCLCL - 70	ns
TLLDV	ALE Low to Valid Data In		507		8TCLCL - 160	ns
TAVDV	Address to Valid Data In		575		9TCLCL - 175	ns
TLLWL	ALE Low to \overline{RD} or \overline{WR} Low	200	300	3TCLCL - 50	3TCLCL + 50	ns
TAVWL	Address Valid to \overline{WR} Low	203		4TCLCL - 130		ns
TQVWX	Address Valid before \overline{WR}	23		TCLCL - 60		ns
TWHQX	Data Hold after \overline{WR}	33		TCLCL - 50		ns
TRLAZ	\overline{RD} Low to Address Float		0		0	ns
TWHLH	\overline{RD} or \overline{WR} High to ALE High	43	123	TCLCL - 40	TCLCL + 40	ns

EXTERNAL PROGRAM MEMORY READ CYCLE



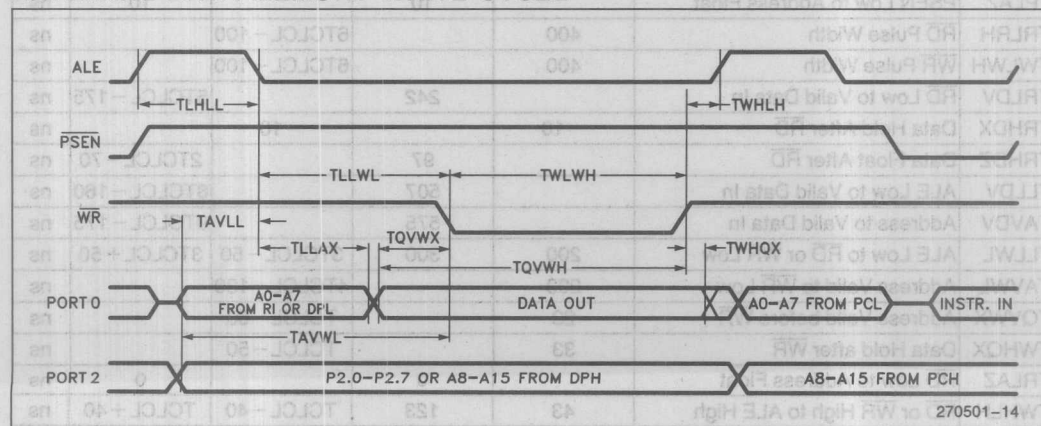
270501-12

EXTERNAL DATA MEMORY READ CYCLE



270501-13

EXTERNAL DATA MEMORY WRITE CYCLE



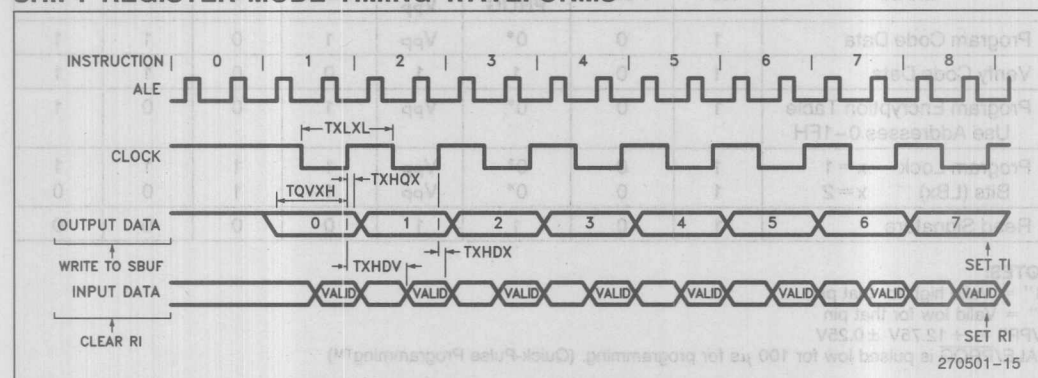
270501-14

SERIAL PORT TIMING - SHIFT REGISTER MODE

Test Conditions: $T_A = -40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL - 133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL - 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL - 133	ns

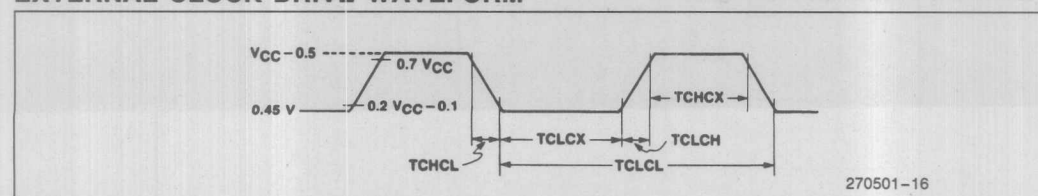
SHIFT REGISTER MODE TIMING WAVEFORMS



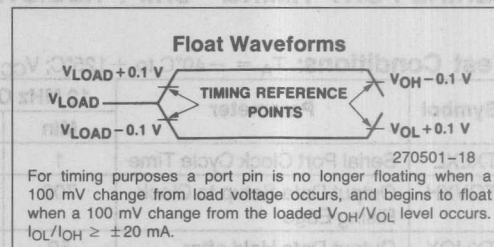
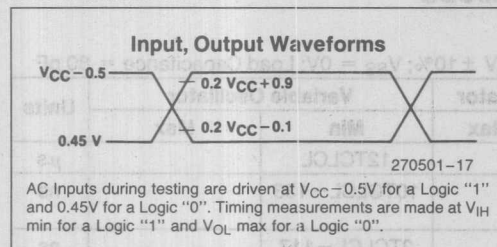
EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORM



A.C. TESTING INPUT



EPROM CHARACTERISTICS

Table 3 shows the logic levels for programming the Program Memory, the Encryption Table, and the Lock Bits and for reading the signature bytes.

Table 3. EPROM Programming Modes

Mode	RST	PSEN	ALE/ PROG	\overline{EA}/V_{PP}	P2.7	P2.6	P3.6	P3.7
Program Code Data	1	0	0*	V_{PP}	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Program Encryption Table Use Addresses 0-1FH	1	0	0*	V_{PP}	1	0	0	1
Program Lock x=1	1	0	0*	V_{PP}	1	1	1	1
Bits (LBx) x=2	1	0	0*	V_{PP}	1	1	0	0
Read Signature	1	0	1	1	0	0	0	0

NOTES:

"1" = Valid high for that pin

"0" = Valid low for that pin

" V_{PP} " = +12.75V \pm 0.25V

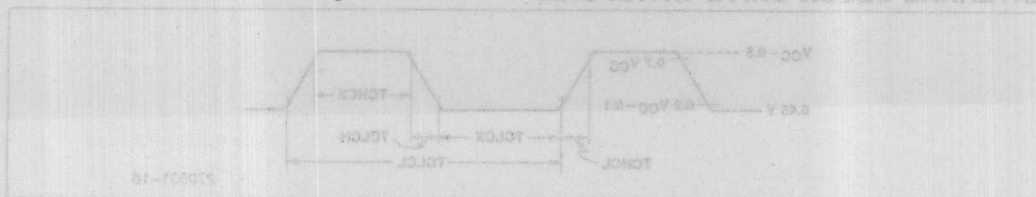
* ALE/PROG is pulsed low for 100 μ s for programming. (Quick-Pulse Programming™)

PROGRAMMING THE EPROM

To be programmed, the part must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal EPROM locations.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0 - P2.4 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 and 3 pins, RST PSEN, and \overline{EA}/V_{PP} should be held at the "Program" levels indicated in Table 2. ALE/PROG is pulsed low to program the code byte into the addressed EPROM location. The setup is shown in Figure 11.

Normally \overline{EA}/V_{PP} is held at logic high until just before ALE/PROG is to be pulsed. Then \overline{EA}/V_{PP} is raised to V_{PP} , ALE/PROG is pulsed low, and then \overline{EA}/V_{PP} is returned to a valid high voltage. The voltage on the \overline{EA}/V_{PP} pin must be at the valid \overline{EA}/V_{PP} high level before a verify is attempted. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

Note that the \overline{EA}/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches.



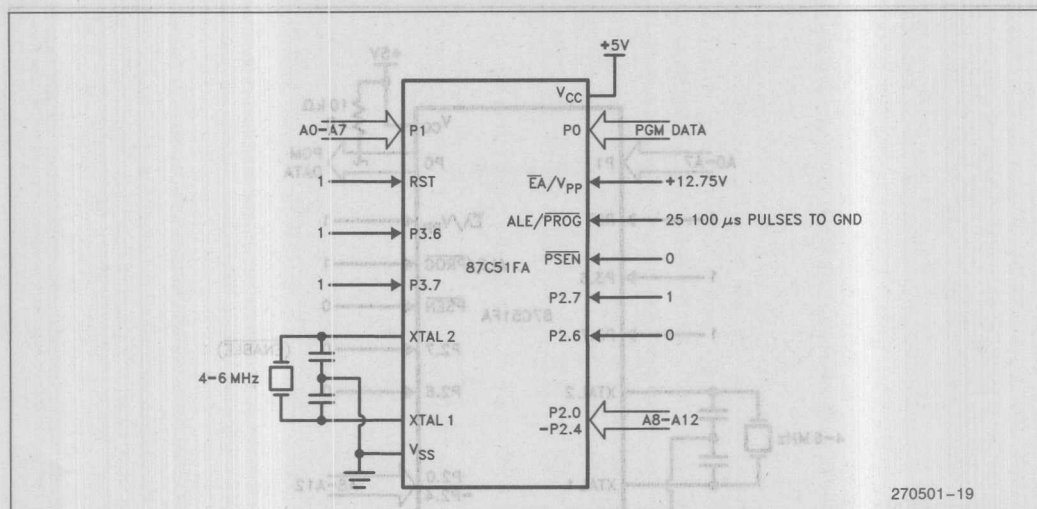


Figure 11. Programming the EPROM

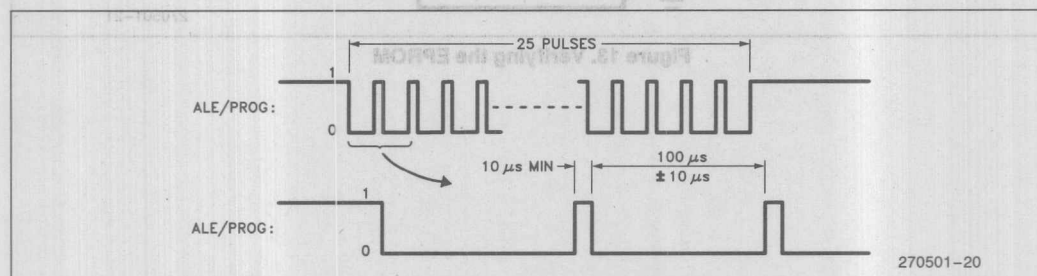


Figure 12. PROG Waveforms

Quick-Pulse Programming™ Algorithm

The 87C51FA can be programmed using the Quick-Pulse Programming™ Algorithm for microcontrollers. The features of the new programming method are a lower V_{pp} (12.75V as compared to 21V) and a shorter programming pulse. It is possible to program the entire 8K Bytes of EPROM memory in less than 25 seconds with this algorithm!

To program the part using the new algorithm, V_{pp} must be $12.75V \pm 0.25V$. ALE/PROG is pulsed low for 100 μs , 25 times as shown in Figure 12. Then, the byte just programmed may be verified. After programming, the entire array should be verified. The Program Lock features are programmed using the same method, but with the setup as shown in Table 2. The only difference in programming Program Lock features is that the Program Lock features cannot be directly verified. Instead, verification of programming is by observing that their features are enabled.

Program Verification (EPROM only)

If the Program Lock Bits have not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0 - P2.4. The other pins should be held at the "Verify" levels indicated in Table 4. The contents of the addressed locations will come out on Port 0. External pullups are required on Port 0 for this operation.

If the Encryption Array in the EPROM has been programmed, the data present at Port 0 will be Code Data XNOR Encryption Data. The user must know the Encryption Array contents to manually "decrypt" the data during verify.

The setup, which is shown in Figure 13, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active-low read strobe.

EPROM Program Lock

The two-level Program Lock system consists of two Program Lock bits and a 32 byte Encryption Array which are used to protect the program memory against software piracy.

Encryption Array (EPROM only)

Within the EPROM array are 32 bytes of Encryption Array that are initially unprogrammed (all 1's). Every time that a byte is addressed during a verify, 5 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NOR'ed (XNOR) with the code byte, creating an Encrypted Verify byte. The algorithm, with the array in the unprogrammed state (all 1's), will return the code in it's original, unmodified form.

Program Lock Bits (EPROM only)

Also included in the EPROM Program Lock scheme are two Program Lock Bits which are programmed as shown in Table 3.

Erasing the EPROM also erases the Encryption Array and the Program Lock Bits, returning the part to full functionality.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values returned are:

(030H) = 89H indicates manufactured by Intel

(031H) = 50H indicates 87C51FA

(031H) = 53H indicates 83C51FA

Erase Characteristics (EPROM only)

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelength shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm. Exposing the EPROM to an ultraviolet lamp of 12,000 μ W/cm rating for 30 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the all EPROM Cells in a 1's state.

Table 4. Program Lock Bits and their Features

Program Lock Bits LB1 LB2		Logic Enabled
U	U	No Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array.)
P	U	MOVX instructions executed from external program memory are disabled from fetching code bytes from internal memory, \overline{EA} is sampled and latched on reset, and further programming of the EPROM is disabled.
P	P	Same as above, but Verify is also disabled
U	P	Reserved for Future Definition

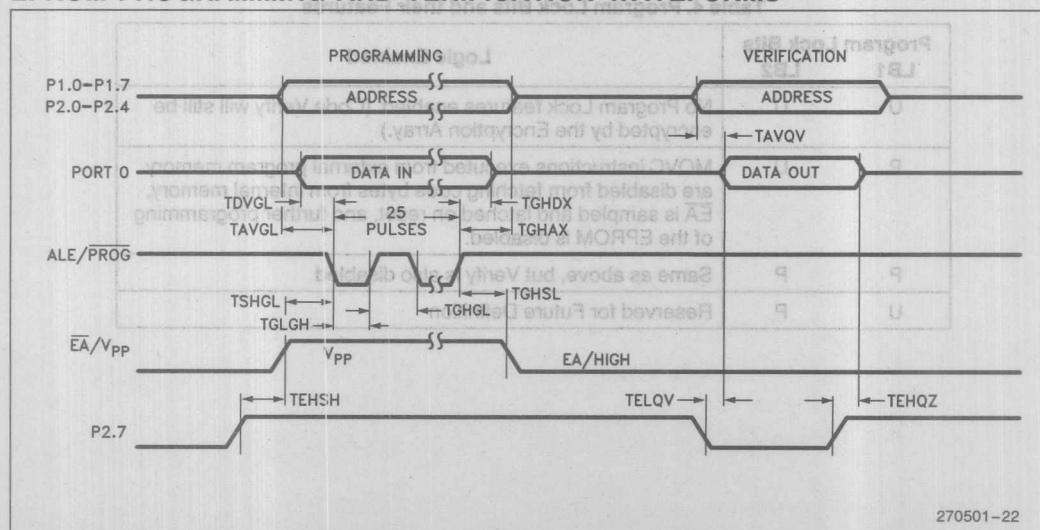
ADVANCED INFORMATION—CONTACT INTEL FOR DESIGN-IN INFORMATION

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

($T_A = 21^\circ\text{C}$ to 27°C ; $V_{CC} = 5V \pm 0.25V$; $V_{SS} = 0V$)

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	12.5	13.0	V
I_{PP}	Programming Supply Current		50	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to PROG Low	48TCLCL		
TGHAX	Address Hold after PROG	48TCLCL		
TDVGL	Data Setup to PROG Low	48TCLCL		
TGHDX	Data Hold after PROG	48TCLCL		
TEHSH	P2.7 (ENABLE) High to V_{PP}	48TCLCL		
TSHGL	V_{PP} Setup to PROG Low	10		μs
TGHSL	V_{PP} Hold after PROG	10		μs
TGLGH	PROG Width	90	110	μs
TAVQV	Address to Data Valid		48TCLCL	
TELQV	ENABLE Low to Data Valid		48TCLCL	
TEHQZ	Data Float after ENABLE	0	48TCLCL	
TGHGL	PROG High to PROG Low	10		μs

EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



270501-22

80C51GB/83C51GB/87C51GB CHMOS SINGLE-CHIP 8-BIT MICROCONTROLLER WITH 8K BYTES USER PROGRAMMABLE EPROM/MASK ROM

- Extended Automotive Temperature Range (-40°C to $+125^{\circ}\text{C}$ Ambient)
- High Performance CHMOS ROM/EPROM
- 8 Channel, 8-bit A/D Converter
- Two 5-channel Programmable Counter Arrays with:
 - High Speed Output,
 - Compare/Capture,
 - Pulse Width Modulator,
 - Watch Dog Timer Capabilities
- 48 Programmable I/O Lines, 42 with Schmitt Trigger Inputs
- Dedicated Watch Dog Timer
- Oscillator Failure Detection
- Three 16-Bit Timer/Counters
- Programmable Serial Channel with:
 - Framing Error Detection
 - Automatic Address Recognition
- Serial Expansion Channel
- Boolean Processor
- Quick Pulse Programming™ Algorithm
- Up/Down Timer/Counter
- Two Level Program Lock System
- 8K On-Chip ROM/EPROM
- 256 Bytes On-Chip Data RAM
- 64K External Program Memory Space
- 64K External Data Memory Space
- 7 Interrupt Sources
- TTL Compatible Logic Levels
- MCS®-51 Instruction SuperSet
- Power Saving Idle and Power Down Modes
- ONCE™ (ON-Circuit Emulation) mode
- Available IN 68-lead Hermetic, Plastic PLCC Packages

(see Packaging Specification, Order #231369)

MEMORY ORGANIZATION

PROGRAM MEMORY: Up to 8K bytes of the program memory can reside in the on-chip ROM/EPROM. In addition the device can address up to 64K of program memory external to the chip.

DATA MEMORY: This microcontroller has a 256 x 8 on-chip RAM. In addition it can address up to 64K bytes of external data memory.

The Intel 87C51GB/83C51GB/80C51GB is a single-chip control oriented microcontroller which is fabricated on Intel's CHMOS III (83C51GB) ROM and CHMOS II-E (87C51GB) EPROM technology. For the remainder of this datasheet, reference to the ROMless (80C51GB), ROM (83C51GB), and EPROM (87C51GB) versions will be denoted as 87C51GB. Being a member of the MCS®-51 family, the 87C51GB uses the same powerful instruction set, has the same architecture, and is pin for pin compatible with the existing MCS-51 products. The 87C51GB is an enhanced version of the 87C51. It's added features make it an even more powerful microcontroller for applications that require Pulse Width Modulation, High Speed I/O, and up/down counting capabilities such as motor control. It also has a more versatile serial channel that facilitates multi-processor communications.

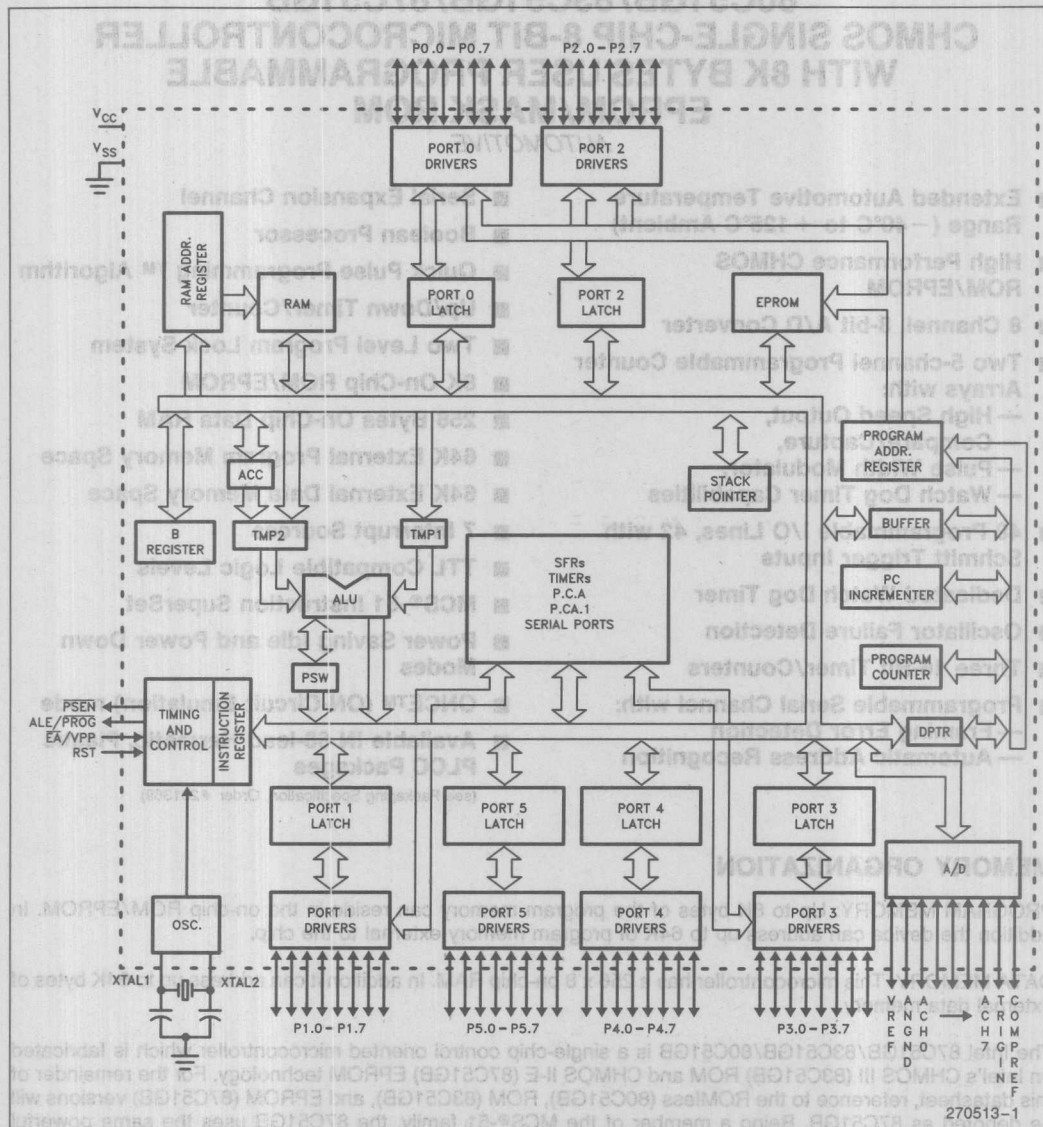


Figure 1. 87C51GB Block Diagram

87C51GB PRODUCT OPTIONS

Intel's extended and automotive temperature range products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to 70°C ambient. With the extended temperature range option, operational characteristics are guaranteed over the temperature range of -40°C to +85°C ambient. For the automotive temperature range option, operational characteristics are guaranteed over the temperature range of -40°C to +125°C ambient.

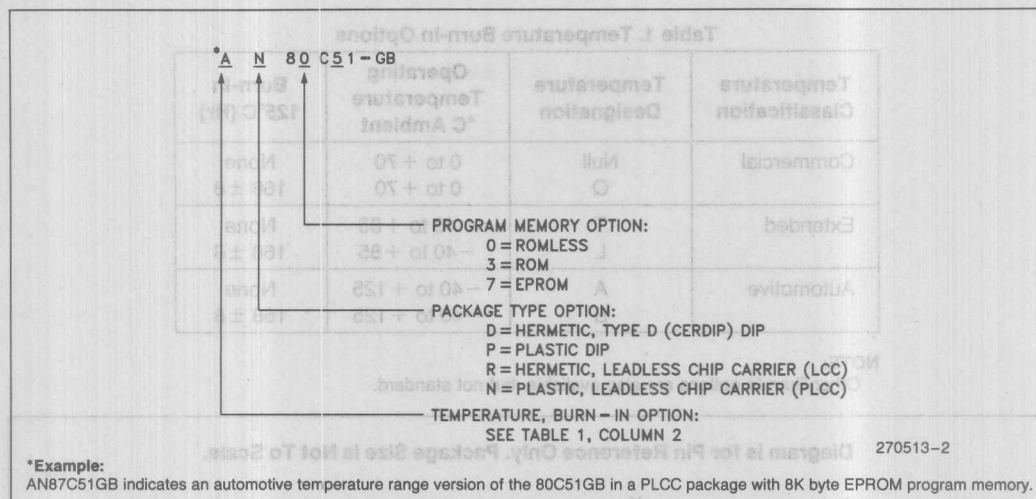


Figure 2. MCS[®]-51 Product Family Nomenclature

The automotive, extended, and commercial temperature versions of the MCS-51 product families are available with or without burn-in options as listed in Table 1.

As shown in Figure 2, temperature, burn-in, and package options are identified by a one- or two-letter prefix to the part number.

PARALLEL I/O PORTS

The 87C51GB contains six 8-bit parallel I/O ports. All six ports are bidirectional and consist of a latch, an output driver, and an input buffer. Many of the port pins have multiplexed I/O and control functions.

Port Pins As Outputs

Port 0 has open drain outputs when it is not serving as the external data bus. The internal pullup is active only when the pin is outputting a logic 1 during external memory access. An external pullup resistor is required on Port 0 when it is serving as an output port.

Ports 1, 2, 3, 4 and 5 have quasi-bidirectional outputs. A strong pullup provides a fast rise time when the pin is set to a logic 1. This pullup turns on for two oscillator periods to drive the pin high and then turns off. The pin is held high by a weak pullup.

Writing the P0, P1, P2, P3, P4 or P5 Special Function Register sets the corresponding port pins. All six port registers are bit addressable.

Port Pins As Inputs

The pins of all six ports are configured as inputs by writing a logic 1 to them. Since Port 0 is an open drain port, it provides a very high input impedance. Since pins of Port 1, 2, 3, 4 and 5 have weak pullups (which are always on), they source a small current when driven low externally. All ports except POA0 have Schmitt trigger inputs with a minimum hysteresis of 0.4 Volts.

Port States During Reset

Ports 0, 1 and 3 reset asynchronously to a one and Ports 2, 4, and 5 reset to a zero asynchronously. Ports 2, 4 and 5 will reset to a one when the reset is done to activate a test mode.

PIN DESCRIPTIONS

The 87C51GB will be packaged in the 68 lead PLCC and CERQUAD package. Its pin assignment is shown in Figure 3.

Table 1. Temperature Burn-In Options

Temperature Classification	Temperature Designation	Operating Temperature °C Ambient	Burn-In 125°C (Hr)
Commercial	Null Q	0 to +70 0 to +70	None 168 ±8
Extended	T L	-40 to +85 -40 to +85	None 168 ±8
Automotive	A B	-40 to +125 -40 to +125	None 168 ±8

NOTE:

Other Burn-In options are also available, but not standard.

Diagram is for Pin Reference Only. Package Size is Not To Scale.

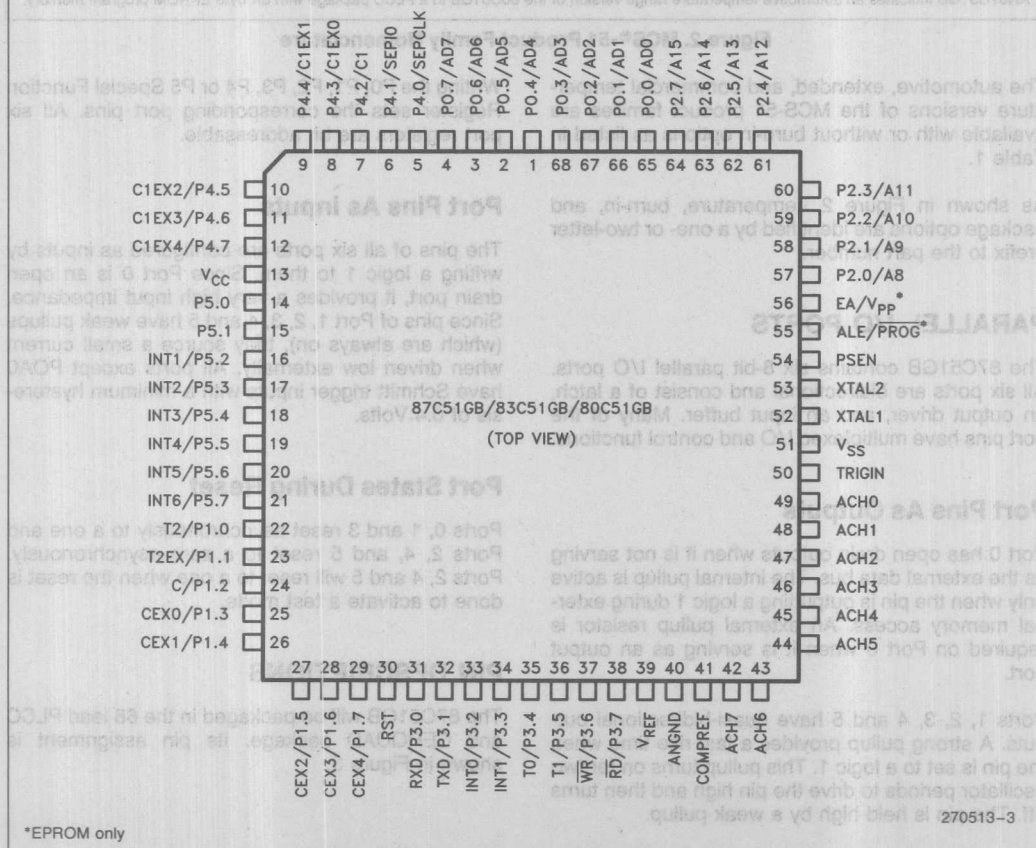


Figure 3. Pin Connections

ALTERNATE PORT FUNCTIONS

Ports 0, 1, 2, 3, 4 and 5 have alternate functions as well as their I/O function as described below.

Port Pin	Alternate Function
P0.0/ADO - P0.7/AD7	Multiplexed Address/Data for External Memory
P1.0/T2	Timer 2 External Clock Input
P1.1/T2EX	Timer 2 Reload/Capture/Direction Control
P1.2/C	PCA External Clock Input
P1.3/CEX0 - P1.7/CEX4	PCA Capture Input, Compare/PWM Output
P2.0/A8 - P2.7/A15	High Byte of Address for External Memory
P3.0/RXD	Serial Port Input
P3.1/TXD	Serial Port Output
P3.2/INT0	External Interrupt
P3.3/INT1	External Interrupt
P3.4/TO	Timer 0 External Clock Input
P3.5/T1	Timer 1 External Clock Input
P3.6/WR	Write Strobe for External Memory
P3.7/RD	Read Strobe for External Memory
P4.0/SEPCLK	Clock Source for Serial Expansion Port
P4.1/SEPDAT	Data I/O for the Serial Expansion Port
P4.2/C1	PCA1 External Clock Input
P4.3/C1EXO - P4.7/C1EX4	PCA1 Capture Input, Compare/PWM Output
P5.2/INT2 - P5.6/INT6	External Interrupt INT2 - INT6

V_{CC}: Supply Voltage.

V_{SS}: Circuit ground.

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. An internal pulldown resistor permits a power-on with only a capacitor connected to V_{CC}.

ALE/PROG: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin (ALE/PROG) is also the program pulse input during EPROM programming for 87C51GB.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that on ALE pulse is skipped during each access to external Data Memory.

Throughout the remainder of this data sheet, ALE will refer to the signal coming out of the ALE/PROG pin, and the pin will be referred to as the ALE/PROG pin.

PSEN: Program Store Enable is the read strobe to external Program Memory.

When the 87C51GB is executing code from external Program Memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external Data Memory.

EA/V_{pp}: External Access enable. EA must be strapped to V_{SS} in order to enable the device to fetch code from external Program Memory locations 0000H to 1FFFFH. Note, however, that if either of the Program Lock bits are programmed, EA will be internally latched on reset.

EA should be strapped to V_{CC} for internal program executions.

This pin also receives the 12.75V programming supply voltage (V_{pp}) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

A/D CONVERTER

The 87C51GB A/D converter will have a resolution of 8 bits and an accuracy of $\pm \frac{1}{2}$ LSB. The conversion time for a single channel will be 20 μ sec at a clock frequency of 16 MHz with the sample and hold function included. Independent supply voltages are provided for the A/D. Also, the A/D will operate both in normal mode or in idle mode.

The A/D has 8 analog input pins; ACH0 (A/D Channel 0) ACH7, 1 reference input pin; COMPREF (COMParison REFerence), 1 control input pin; TRIGIN (TRIGger IN), and 2 power pins; VREF (Voltage REFerence) and ANGND (ANalog GROUND). In addition, the A/D has 8 conversion result registers; AD0 (A/D result for channel 0) AD7, 1 comparison result register; ACMP (Analog CoMParison), and 1 control register; ACON (A/D CControl).

The control bit ACE (A/D Conversion Enable) in ACON controls whether the A/D is in operation or not. ACE=0 idles the A/D. ACE=1 enables A/D conversion. The control bit AIM (A/D Input Mode) in ACON controls the mode of channel selection. AIM = 0 is the scan mode, and AIM = 1 is the select mode. The result registers AD4 .. AD7 always contain the result of a conversion from the corresponding channels ACH4 .. ACH7. However, the result registers AD0 .. AD3 depend on this mode. In the scan mode, AD0 .. AD3 contain the values from ACH0 .. ACH3. In the select mode, one of the four channels ACH0 .. ACH3 is converted four times, and the four values are stored sequentially in locations AD0 .. AD3. Its channel is selected by bits ACS1 and ACS0 (A/D Channel Select 1 and 0) in ACON.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of a inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 4. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 floats, as shown in Figure 5. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the data sheet must be observed.

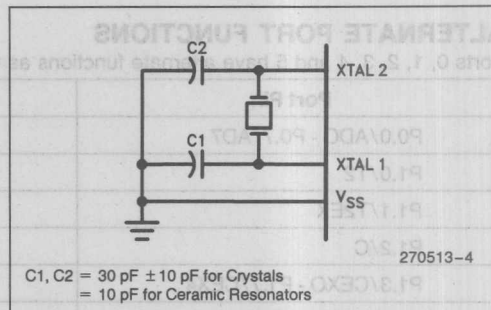


Figure 4. Oscillator Connections

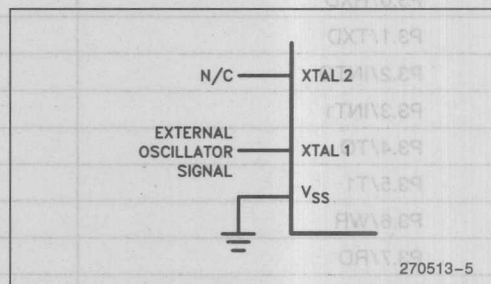


Figure 5. External Clock Drive Configuration

IDLE MODE

The user's software can invoke the Idle Mode. When the microcontroller is in this mode, power consumption is reduced. The Special Function Registers and the onboard RAM retain their values during idle, but the processor stops executing instructions. Idle Mode will be exited if the chip is reset or if an enabled interrupt occurs. The PCA timer/counter can optionally be left running or paused during Idle Mode.

POWER DOWN MODE

To save even more power, a Power Down mode can be invoked by software. In this mode, the oscillator is stopped and the instruction that invoked Power Down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power Down mode is terminated.

On the 87C51GB either a hardware reset or an external interrupt can cause an exit from Power Down. Reset redefines all the SFRs but does not change the on-chip RAM. An external interrupt allows both the SFRs and on-chip RAM to retain their values. The interrupt must be enabled and configured as level sensitive. To properly terminate Power Down the reset or external interrupt should not be execut-

ed before V_{CC} is restored to its normal operating level, and must be held active long enough for the oscillator to restart and stabilize.

DESIGN CONSIDERATION

- Ambient light is known to affect the internal RAM contents during operation. If the 87C51GB application requires the part to be run under ambient lighting, an opaque label should be placed over the window to exclude light.
- When the idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

ONCE™ MODE

The ONCE ("On-Circuit Emulation") Mode facilitates testing and debugging of systems using the 87C51GB without the 87C51GB having to be removed from the circuit. The ONCE Mode is invoked by:

- Pull ALE low while the device is in reset and \overline{PSEN} is high;
- Hold ALE low as RST is deactivated.

While the device is in ONCE Mode, the Port 0 pins go into a float state, and the other port pins and ALE and \overline{PSEN} are weakly pulled high. The oscillator circuit remains active. While the 87C51GB is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored when a normal reset is applied.

Table 2. Status of the External Pins during Idle and Power Down

Mode	Program Memory	ALE	\overline{PSEN}	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

NOTE:

For more detailed information on the reduced power modes refer to current Embedded Controller Handbook, and Application Note AP-252, "Designing with the 80C51BH."

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias -40°C to $+125^{\circ}\text{C}$
 Storage Temperature -65°C to $+150^{\circ}\text{C}$
 Voltage on EA/ V_{PP} Pin to V_{SS} 0V to $+13.0\text{V}$ *
 Voltage on Any Other Pin to V_{SS} -0.5V to $+6.5\text{V}$
 Power Dissipation 1.5W
 (Based on Package heat transfer limitations, not device power consumption)

*EPROM only.

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

ADVANCED INFORMATION - Contact Intel for Design-In Information

DC CHARACTERISTICS $T_A = -40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ case temperature; $V_{CC} = 4.5\text{V}$ to 5.5V ; $V_{SS} = 0\text{V}$

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
V_{T+}	High-going Threshold (except XTAL1, RST, POA0)	1.10	1.50	2.0	V	
V_{T-}	Low-going Threshold (except XTAL1, RST, POA0)	0.60	0.90	1.2	V	
V_{HYS}	Hysteresis ($V_{T+} - V_{T-}$) (except XTAL1, RST, POA0)	0.40	0.60		V	
V_{IL}	Input Low Voltage (XTAL1, RST, POA0)	-0.5		$0.2 V_{CC} - 0.1$	V	
V_{IH}	Input High Voltage (XTAL1, RST)	$0.7 V_{CC}$		$V_{CC} + 0.5$	V	
V_{IH1}	Input High Voltage (Port 0)	$0.2 V_{CC} + 0.9$		$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage (Ports 1,2,3,4,5)			0.45	V	$I_{OL} = 1.6\text{ mA}$
V_{OL1}	Output Low Voltage (Port0, ALES, PSEN)			0.45	V	$I_{OL} = 3.2\text{ mA}$
V_{OH}	Output High Voltage (Ports 1,2,3,4,5)	2.4 $0.9 V_{CC}$			V	$I_{OH} = -60\text{ }\mu\text{A}$ $I_{OH} = -10\text{ }\mu\text{A}$
V_{OH1}	Output High Voltage (Port0 in External Bus Mode, ALE, PSEN)	2.4 $0.9 V_{CC}$			V	$I_{OH} = -800\text{ }\mu\text{A}$ $I_{OH} = -80\text{ }\mu\text{A}$
I_{IL}	Logical 0 Input Current (Ports 1,2,3,4,5)			-50	μA	$V_{IN} = 0.45\text{V}$
I_{TL}	Logical 1-to-0 transition current (Ports 1,2,3,4,5)			-650	μA	$V_{IN} = 2.0\text{V}$
I_{LI}	Input Leakage Current (Port 0, \bar{EA})			± 10	μA	$0.45 < V_{IN} < V_{CC}$
RRST	RST Pulldown Resistor	40		125	$\text{k}\Omega$	
C_{IO}	Pin Capacitance			10	pF	freq = 1 MHz $T_A = 25^{\circ}\text{C}$
I_{PD}	Power Down Current			200	μA	(2)
I_{DL}	Idle Mode Current			10	mA	(3)
I_{CC}	Operating Current			40	mA	(4)

NOTES:

1: Typical values are obtained using $V_{CC} = 5.0\text{V}$ and $T_A = 25^{\circ}\text{C}$.

2: Power Down Current is measured with all output pins disconnected, $EA = \text{PORT0} = V_{CC}$, XTAL2 N.C., RST = V_{SS} .

3: Idle I_{CC} is measured with all output pins disconnected, XTAL1 driven with TCHLCH, TCHLCH = 10 ns, $V_{IL} V_{SS} + 0.5\text{V}$, $V_{IH} = V_{CC} - 0.5\text{V}$, XTAL2 N.C., $EA = \text{PORT0} = V_{CC}$, RST = V_{SS} , internal clock to PCA gated off.

4: I_{CC} is measured with all output pins disconnected, XTAL1 driven with TCLCH, TCLCH = 10 ns, $V_{IL} = V_{SS} + 0.5\text{V}$, $V_{IH} = V_{CC} - 0.5\text{V}$, XTAL2 N.C., $EA = \text{RST} = \text{PORT0} = V_{CC}$.

AC SPECIFICATIONS

$T_A = -40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ case temperature, $V_{CC} = 4.5$ to 5.5 V, Load Capacitance on Port 0, ALE, and PSEN = 100 pF, Load Capacitance on all other outputs = 80 pF

Advanced Information - Contact Intel for Design-In information.

EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS

Symbol	Parameter	12 MHz Osc.		Variable Osc.		Units
		Min	Max	Min	Max	
1/TCLCL	Osc. Freq.			3.5	12	MHz
TLHLL	ALE Pulse Width	112		2TCLCL - 55		ns
TAVLL	ADDR Valid To ALE Low	13		TCLCL - 70		ns
TLLAX	ADDR Hold After ALE Low	33		TCLCL - 50		ns
TLLIV	ALE Low to Valid Instr. IN		218		4TCLCL - 115	ns
TLLPL	ALE LOW to PSEN LOW	28		TCLCL - 55		ns
TPLPH	PSEN Pulse Width	190		3TCLCL - 60		ns
TPLIV	PSEN Low to Valid Instr. In		130		3TCLCL - 120	ns
TPXIX	Input Instr. Hold After PSEN	0		0		ns
TPXIZ	Input Instr. Float After PSEN		43		TCLCL - 40	ns
TAVIV	ADDR To Valid Instr. In		297		5TCLCL - 120	ns
TPLAZ	PSEN Low to ADDR Float		25		25	ns
TRLRH	RD PULSE WIDTH	400		6TCLCL - 100		ns
TWLWH	WR PULSE WIDTH	400		6TCLCL - 100		ns
TRLDV	RD Low to Valid Data In		232		5TCLCL - 185	ns
TRHDX	Data Hold After RD	0		0		ns
TRHDZ	Data Float After RD		82		2TCLCL - 85	ns
TLLDV	ALE Low to Valid Data In		496		8TCLCL - 170	ns
TAVDV	ADDR To Valid Data In		565		9TCLCL - 185	ns
TLLWL	ALE Low to RD or WR Low	185	315	3TCLCL - 65	3TCLCL + 65	ns
TAVWL	ADDR TO RD or WR Low	188		4TCLCL - 145		ns
TQVWX	Data Valid to WR Transition	8		TCLCL - 75		ns
TWHGX	Data Hold After WR	18		TCLCL - 65		ns
TRLAZ	RD Low to Addr Float		0		0	ns
TWHLH	RD or WR High to ALE High	18	148	TCLCL - 65	TCLCL + 65	ns

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a T (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for:

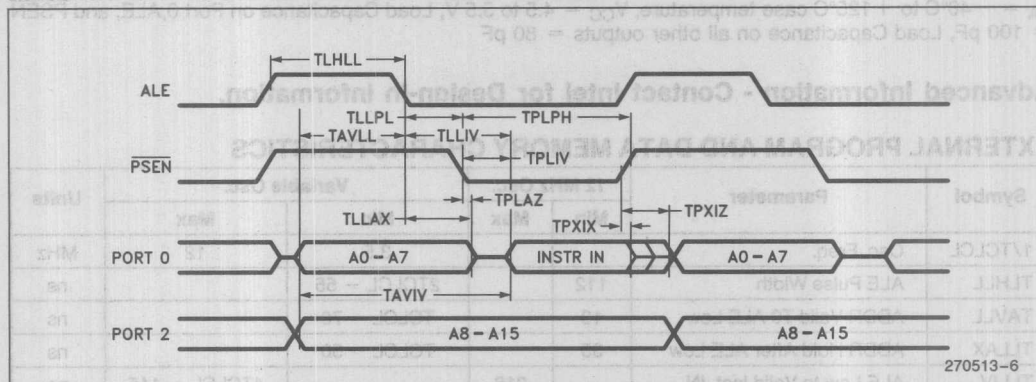
- A: Address
- C: Clock
- D: Input Data
- H: Logic level HIGH
- I: Instruction (program memory contents)

- L: Logic level LOW, or ALE
- P: PSEN
- Q: Output Data
- R: RD signal
- T: Time
- V: Valid
- W: WR signal
- X: No longer a valid logic level
- Z: Float

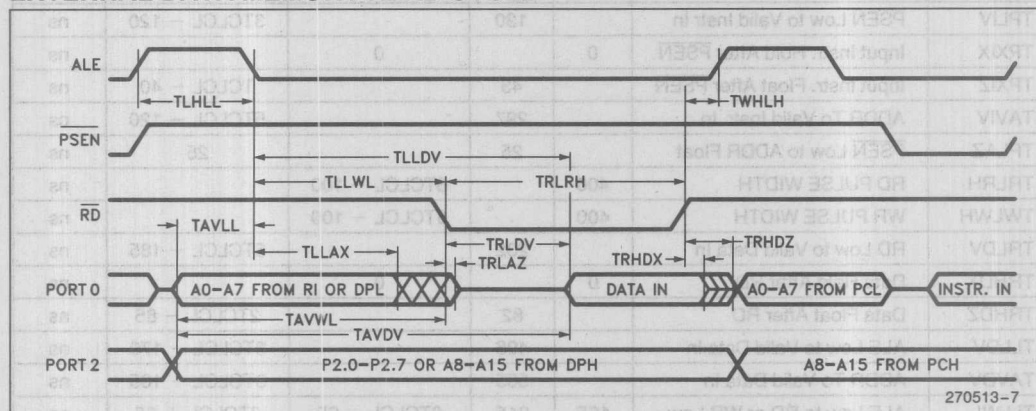
For example,

TAVLL = Time from Address Valid to ALE Low
TLLPL = Time from ALE Low to PSEN Low

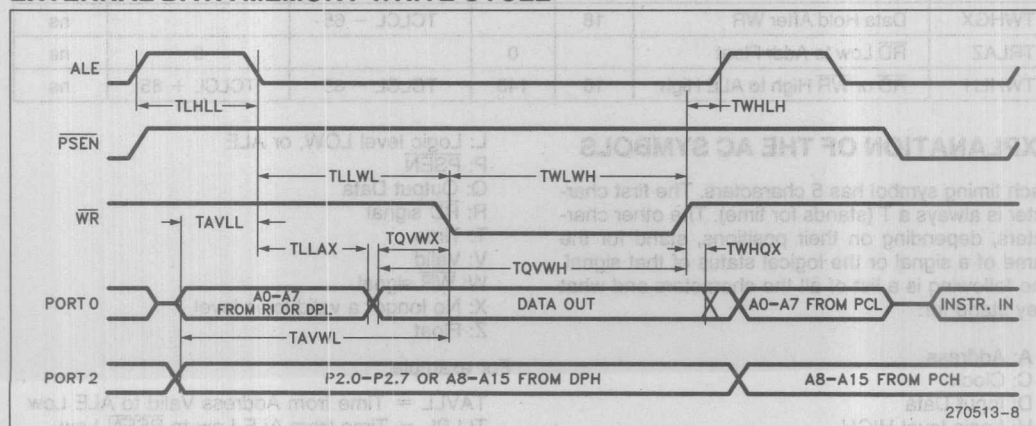
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE

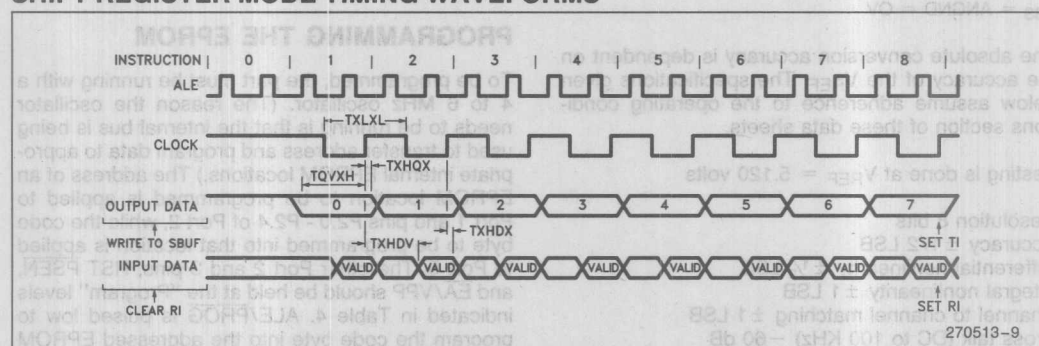


SERIAL PORT TIMING - SHIFT REGISTER MODE

Test Conditions: TA = -40°C to 125°C; V_{CC} = 5V ± 10%; V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL - 133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL - 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL - 133	ns

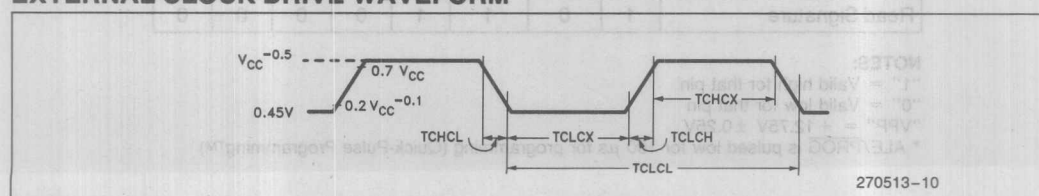
SHIFT REGISTER MODE TIMING WAVEFORMS



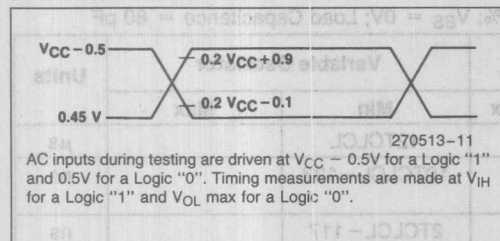
EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

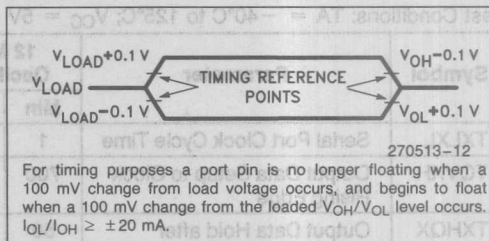
EXTERNAL CLOCK DRIVE WAVEFORM



A.C. TESTING INPUT



Input, Output Waveforms



Float Waveforms

A/D CONVERTER SPECIFICATIONS

$T_A = -40^\circ C$ to $+125^\circ C$ case temperature
 $V_{CC} = 4.5$ TO 5.5 V
 $V_{REF} = 4.5$ TO 5.5 V
 $V_{SS} = ANGND = OV$

The absolute conversion accuracy is dependent on the accuracy of the V_{REF} . The specifications given below assume adherence to the operating conditions section of these data sheets.

Testing is done at $V_{REF} = 5.120$ volts

Resolution 8 bits
 Accuracy $\pm 1/2$ LSB
 Differential nonlinearity $\pm 1/2$ LSB
 Integral nonlinearity ± 1 LSB
 Channel to channel matching ± 1 LSB
 Cross talk (DC to 100 KHz) -60 dB

EPROM CHARACTERISTICS

Table 3 shows the logic levels for programming the Program Memory, the Encryption Table, and the Lock Bits and for reading the signature bytes.

PROGRAMMING THE EPROM

To be programmed, the part must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal EPROM locations.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0 - P2.4 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 and 3 pins, RST PSEN, and EA/VPP should be held at the "Program" levels indicated in Table 4. ALE/PROG is pulsed low to program the code byte into the addressed EPROM location. The setup is shown in Figure 10.

Table 3. EPROM Programming Modes

Mode	RST	PSEN	ALE/ PROG	EA/ VPP	P2.7	P2.6	P3.6	P3.7
Program Code Data	1	0	0*	VPP	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Program Encryption Table Use Addresses 0-3F	1	0	0*	VPP	1	0	0	1
Program Lock x = 1	1	0	0*	VPP	1	1	1	1
Bits (LBx) x = 2	1	0	0*	VPP	1	1	0	0
Read Signature	1	0	1	1	0	0	0	0

NOTES:

"1" = Valid high for that pin

"0" = Valid low for that pin

"VPP" = $+12.75V \pm 0.25V$

* ALE/PROG is pulsed low for 100 μs for programming (Quick-Pulse Programming™)

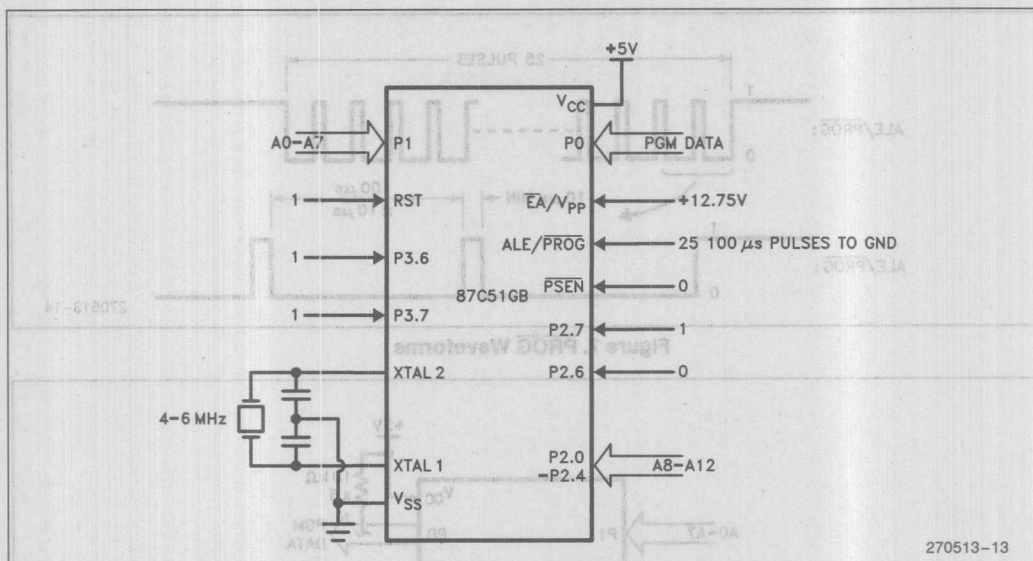


Figure 6. Programming the EPROM

Normally \overline{EA}/V_{PP} is held at logic high until just before ALE/PROG is to be pulsed. Then \overline{EA}/V_{PP} is raised to V_{PP} , ALE/PROG is pulsed low, and then \overline{EA}/V_{PP} is returned to a valid high voltage. The voltage on the \overline{EA}/V_{PP} pin must be at the valid \overline{EA}/V_{PP} high level before a verify is attempted. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

Note that the \overline{EA}/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches.

Quick-Pulse Programming™ Algorithm

The 87C51GB (EPROM only) can be programmed using the Quick-Pulse Programming™ Algorithm for microcontrollers. The features of the new programming method are a lower V_{PP} (12.75V as compared to 21V) and a shorter programming pulse. It is possible to program the entire 8K Bytes of EPROM memory in less than 25 seconds with this algorithm!

To program the part using the new algorithm, V_{PP} must be 12.75V ± 0.25 V. ALE/PROG is pulsed low for 100 μ s, 25 times as shown in Figure 7. Then, the byte just programmed may be verified. After programming, the entire array should be verified. The

Program Lock features are programmed using the same method, but with the setup as shown in Table 4. The only difference in programming Program Lock features is that the Program Lock features cannot be directly verified. Instead, verification of programming is by observing that their features are enabled.

Program Verification (EPROM only)

If the Program Lock Bits have not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read applies to Port 1 and pins P2.0 - P2.4. The other pins should be held at the "Verify" levels indicated in Table 4. The contents of the addressed locations will come out on Port 0. External pullups are required on Port 0. External pullups are required on Port 0 for this operation.

If the Encryption Array in the EPROM has been programmed, the data present at Port 0 will be Code Data XNOR Encryption Data. The user must know the Encryption Array contents to manually "unencrypt" the data during verify.

The setup, which is shown in Figure 8 is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active-low read strobe.

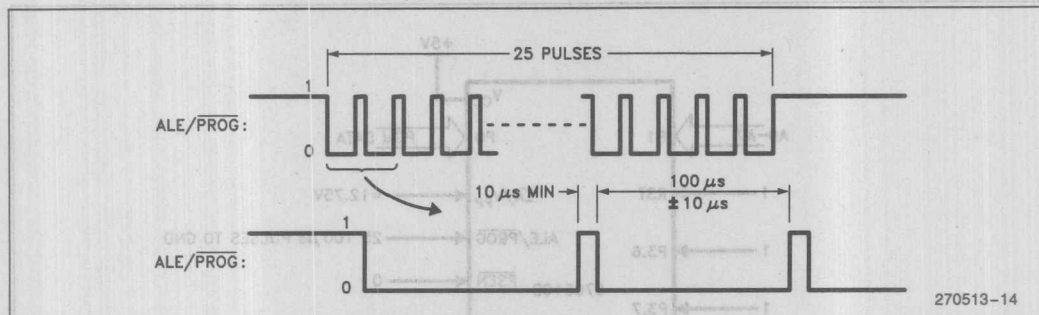


Figure 7. PROG Waveforms

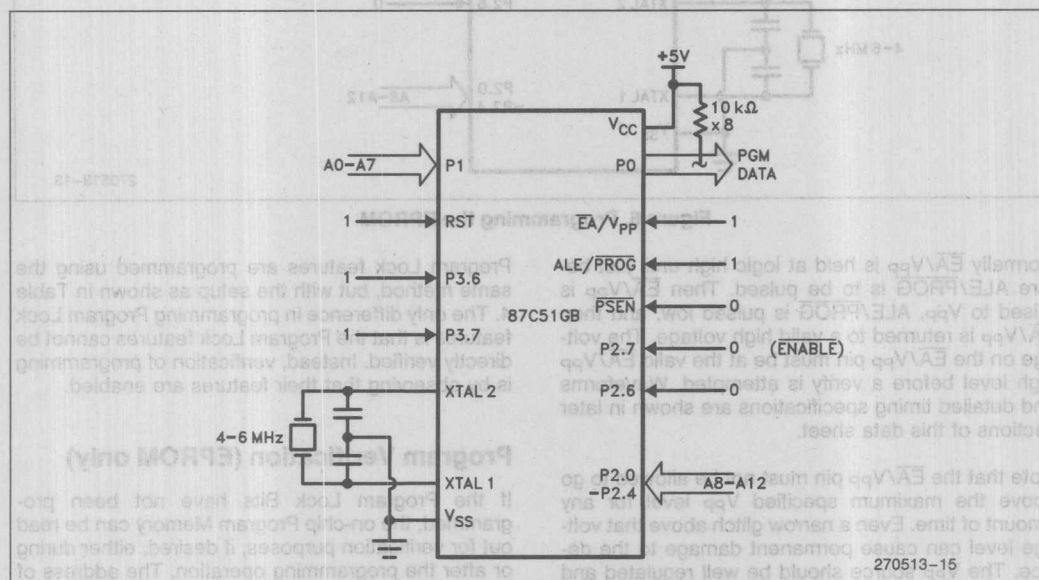


Figure 8. Verifying the EPROM

Table 4. Program Lock Bits and their Features

Program Lock Bits		Logic Enabled
LB1	LB2	
U	U	No Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array.)
P	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled.
P	P	Same as above, but Verify is also disabled.
U	P	Reserved for Future Definition.

EPROM Program Lock

The two-level Program Lock system consists of two Program Lock bits and a 32 byte Encryption Array which are used to protect the program memory against software piracy.

Encryption Array (EPROM only)

Within the EPROM array are 64 bytes of Encryption Array that are initially unprogrammed (all 1's). Every time that a byte is addressed during a verify, 6 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NOR'ed (XNOR) with the code byte, creating an Encrypted XNOR byte. The algorithm, with the array in the unprogrammed state (all 1's), will return the code in it's original, unmodified form.

Program Lock Bits (EPROM only)

Also included in the EPROM Program Lock scheme are two Program Lock bits which provide program Program Lock as shown in Table 4.

Erasing the EPROM also erases the Encryption Array and the Program Lock bits, returning the part to full functionality.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values returned are:

(030H) = 89H indicates manufactured by Intel
(031H) = 5AH indicates 87C51GB
(031H) = 5BH indicates 83C51GB

Erase Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelength shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm. Exposing the EPROM to an ultraviolet lamp of 12,000 uW/cm rating for 30 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the all EPROM Cells in a 1's state.

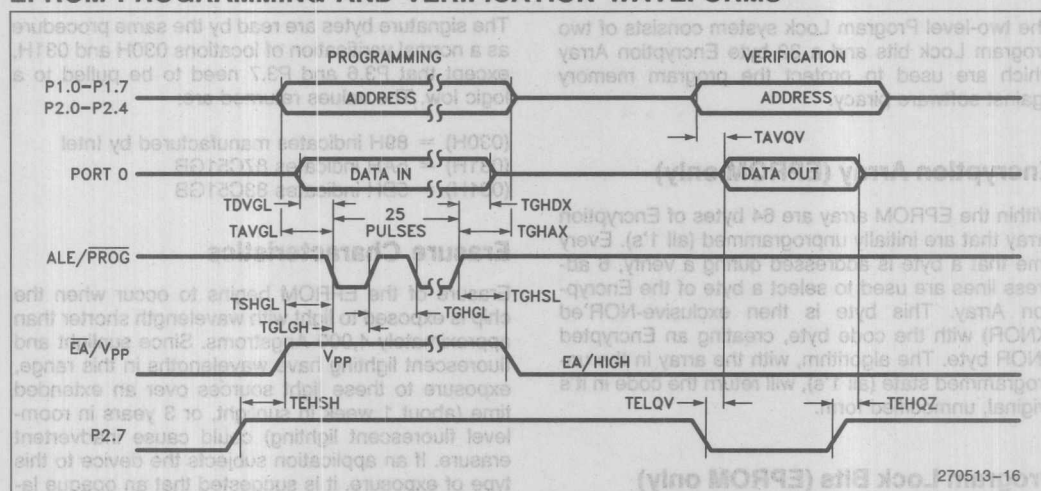
EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

(T_A = 21°C to 27°C; V_{CC} = 5V ± 10%; V_{SS} = 0V)

ADVANCED INFORMATION. CONTACT INTEL FOR DESIGN-IN INFORMATION.

Symbol	Parameter	Min	Max	Units
V _{PP}	Programming Supply Voltage	12.5	13.0	V
IPP	Programming Supply Current		50	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHAX	Address Hold after $\overline{\text{PROG}}$	48TCLCL		
TDVGL	Data Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHDX	Data Hold after $\overline{\text{PROG}}$	48TCLCL		
TEHSH	P2.7 (ENABLE) High to V _{PP}	48TCLCL		
TSHGL	V _{PP} Setup to $\overline{\text{PROG}}$ Low	10		μs
TGHSL	V _{PP} Hold after $\overline{\text{PROG}}$	10		μs
TGLGH	$\overline{\text{PROG}}$ Width	95	105	μs
TAVQV	Address to Data Valid		48TCLCL	
TELQV	ENABLE Low to Data Valid		48TCLCL	
TEHQZ	Data Float after ENABLE	0	48TCLCL	
TGHGL	$\overline{\text{PROG}}$ High to $\overline{\text{PROG}}$ Low	10		μs

EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



The recommended erasure procedure is exposure to ultraviolet light (2537 Angstroms) to an intensity of at least 15 W/sec/cm. Exposing the EPROM to an ultraviolet lamp of 12,000 W/cm² rating for 30 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the all EPROM Cells in a 1's state.

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

(TA = 21°C to 27°C; VCC = 5V ± 10%; Vss = 0V)

ADVANCED INFORMATION. CONTACT INTEL FOR DESIGN INFORMATION.

Symbol	Parameter	Min	Max	Units
Vpp	Programming Supply Voltage	12.5	13.0	V
Ipp	Programming Supply Current	50		mA
1/TCCL	Oscillator Frequency	4	8	MHz
TAVGL	Address Setup to PROG Low	48TCL		
TGHAX	Address Hold after PROG	48TCL		
TDVGL	Data Setup to PROG Low	48TCL		
TGHDX	Data Hold after PROG	48TCL		
TEHSH	P2.7 (ENABLE) High to Vpp	48TCL		
TSHGL	Vpp Setup to PROG Low	10		ns
TGHSL	Vpp Hold after PROG	10		ns
TGLGH	PROG Width	25	105	ns
TAVQV	Address to Data Valid	48TCL		
TELQV	ENABLE Low to Data Valid	48TCL		
TEHQZ	Data Float after ENABLE	0		
TGHGL	PROG High to PROG Low	10		ns

When the members of the MCS-48 and 8051 families. The 8751 contains 4K bytes of EPROM program memory fabricated on-chip, while the 8051 replaces the EPROM with 4K bytes of lower-cost mask-programmed ROM. The 8051 has no program memory on-chip; instead, it accesses up to 64K bytes of program memory from external memory. Otherwise, the three new family members are identical. Throughout this Note, the term "8051" will represent all members of the 8051 Family, unless specifically stated otherwise.

The CPU in each microcomputer is one of the industry's fastest and most efficient for numerical calculations on byte operands. But controllers often deal with bits, not bytes, in the real world, which contacts can only be open or closed, indicators should be either lit or dark, motors are either turned on or off, and so forth. For such control situations the most significant aspect of the MCS-51 architecture is its complete hardware support for one-bit or Boolean variables (named in honor of Mathematician George Boole) as a separate data type.

The 8051 incorporates a number of special features which support the direct manipulation and testing of individual bits and allow the use of single-bit variables in performing logical operations. Taken together, these features are referred to as the MCS-51 Boolean Processing Capabilities. While the bit-processing capabilities alone would be

Using the Intel MCS®-51 Boolean Processing Capabilities

Many concepts embodied by the Boolean Processor will certainly be new even to experienced microcomputer system designers. The purpose of this Application Note is to explain these concepts and show how they are used.

For detailed information on these parts refer to the Intel Microcontroller Handbook, order number 210915. The instruction set, assembly language, and use of the 8051 assembler (ASM51) are further described in the MCS-51 Macro Assembler User's Guide for DOS Systems, order number 122753.

Table 1. Features of Intel's Single-Chip Microcomputers

EPROM Program Memory	ROM Program Memory	External Program Memory	Program Memory (in Program Memory)	Data Memory	instr. Cycle Time	Input/Output Pins	Interrupt Sources	Reg. Banks
8748	8048	8038	4K 4K	128	1.38 μ s	27	2	2
8751	8051	8031	4K 84K	128	1.0 μ s	32	2	4

JOHN WHARTON
MICROCONTROLLER APPLICATIONS

The Intel microcontroller family now has three new members: the Intel® 8031, 8051, and 8751 single-chip microcomputers. These devices, shown in Figure 1, will allow whole new classes of products to benefit from recent advances in integrated electronics. Thanks to Intel's new HMOS technology, they provide larger program and data memory spaces, more flexible I/O and peripheral capabilities, greater speed, and lower system cost than any previous-generation single-chip microcomputer.

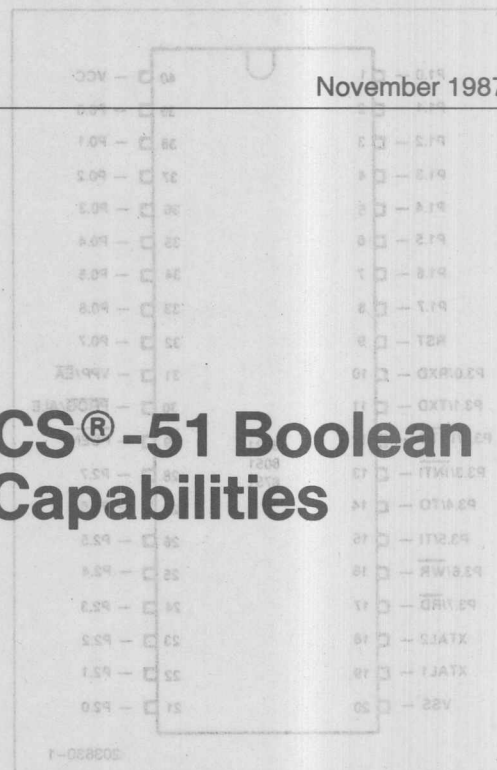


Figure 1. 8051 Family Pinout Diagram

1.0 INTRODUCTION

The Intel microcontroller family now has three new members: the Intel® 8031, 8051, and 8751 single-chip microcomputers. These devices, shown in Figure 1, will allow whole new classes of products to benefit from recent advances in Integrated Electronics. Thanks to Intel's new HMOS technology, they provide larger program and data memory spaces, more flexible I/O and peripheral capabilities, greater speed, and lower system cost than any previous-generation single-chip microcomputer.

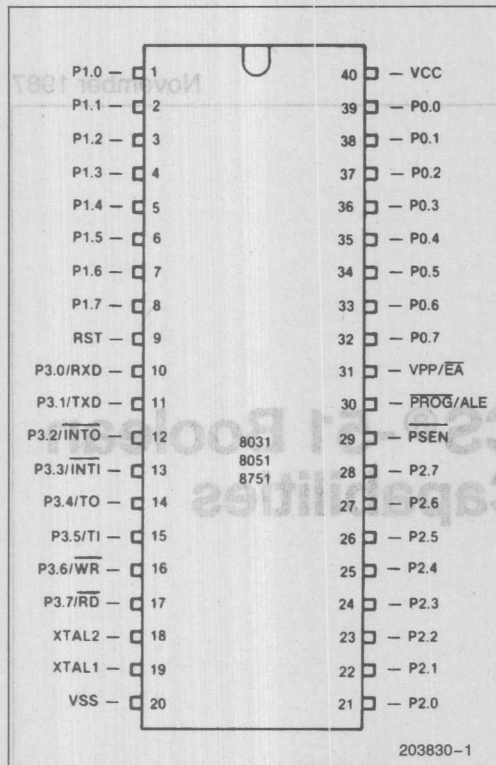


Figure 1. 8051 Family Pinout Diagram

Table 1. Features of Intel's Single-Chip Microcomputers

EPROM Program Memory	ROM Program Memory	External Program Memory	Program Memory (Int/Max)	Data Memory (Bytes)	Instr. Cycle Time	Input/Output Pins	Interrupt Sources	Reg. Banks
8748	8048	8035	1K 4K	64	2.5 μ s	27	2	2
—	8049	8039	2K 4K	128	1.36 μ s	27	2	2
8751	8051	8031	4K 64K	128	1.0 μ s	32	5	4

Table 1 summarizes the quantitative differences between the members of the MCS®-48 and 8051 families. The 8751 contains 4K bytes of EPROM program memory fabricated on-chip, while the 8051 replaces the EPROM with 4K bytes of lower-cost mask-programmed ROM. The 8031 has no program memory on-chip; instead, it accesses up to 64K bytes of program memory from external memory. Otherwise, the three new family members are identical. Throughout this Note, the term "8051" will represent all members of the 8051 Family, unless specifically stated otherwise.

The CPU in each microcomputer is one of the industry's fastest and most efficient for numerical calculations on byte operands. But controllers often deal with bits, not bytes: in the real world, switch contacts can only be open or closed, indicators should be either lit or dark, motors are either turned on or off, and so forth. For such control situations the most significant aspect of the MCS®-51 architecture is its complete hardware support for one-bit, or *Boolean* variables (named in honor of Mathematician George Boole) as a separate data type.

The 8051 incorporates a number of special features which support the direct manipulation and testing of individual bits and allow the use of single-bit variables in performing logical operations. Taken together, these features are referred to as the MCS-51 *Boolean Processor*. While the bit-processing capabilities alone would be adequate to solve many control applications, their true power comes when they are used in conjunction with the microcomputer's byte-processing and numerical capabilities.

Many concepts embodied by the Boolean Processor will certainly be new even to experienced microcomputer system designers. The purpose of this Application Note is to explain these concepts and show how they are used.

For detailed information on these parts refer to the *Intel Microcontroller Handbook*, order number 210918. The instruction set, assembly language, and use of the 8051 assembler (ASM51) are further described in the *MCS®-51 Macro Assembler User's Guide for DOS Systems*, order number 122753.

2.0 BOOLEAN PROCESSOR OPERATION

The Boolean Processing capabilities of the 8051 are based on concepts which have been around for some time. Digital computer systems of widely varying designs all have four functional elements in common (Figure 2):

- a central processor (CPU) with the control, timing, and logic circuits needed to execute stored instructions;
- a memory to store the sequence of instructions making up a program or algorithm;
- data memory to store variables used by the program; and
- some means of communicating with the outside world.

The CPU usually includes one or more accumulators or special registers for computing or storing values during program execution. The instruction set of such a processor generally includes, at a minimum, operation classes to perform arithmetic or logical functions on program variables, move variables from one place to another, cause program execution to jump or conditionally branch based on register or variable states, and instructions to call and return from subroutines. The program and data memory functions sometimes share a single memory space, but this is not always the case. When the address spaces are separated, program and data memory need not even have the same basic word width.

A digital computer's flexibility comes in part from combining simple fast operations to produce more com-

plex (albeit slower) ones, which in turn link together eventually solving the problem at hand. A four-bit CPU executing multiple precision subroutines can, for example, perform 64-bit addition and subtraction. The subroutines could in turn be building blocks for floating-point multiplication and division routines. Eventually, the four-bit CPU can simulate a far more complex "virtual" machine.

In fact, *any* digital computer with the above four functional elements can (given time) complete *any* algorithm (though the proverbial room full of chimpanzees at word processors might first re-create Shakespeare's classics and this Application Note)! This fact offers little consolation to product designers who want programs to run as quickly as possible. By definition, a real-time control algorithm *must* proceed quickly enough to meet the preordained speed constraints of other equipment.

One of the factors determining how long it will take a microcomputer to complete a given chore is the number of instructions it must execute. What makes a given computer architecture particularly well- or poorly-suited for a class of problems is how well its instruction set matches the tasks to be performed. The better the "primitive" operations correspond to the steps taken by the control algorithm, the lower the number of instructions needed, and the quicker the program will run. All else being equal, a CPU supporting 64-bit arithmetic directly could clearly perform floating-point math faster than a machine bogged-down by multiple-precision subroutines. In the same way, direct support for bit manipulation naturally leads to more efficient programs handling the binary input and output conditions inherent in digital control problems.

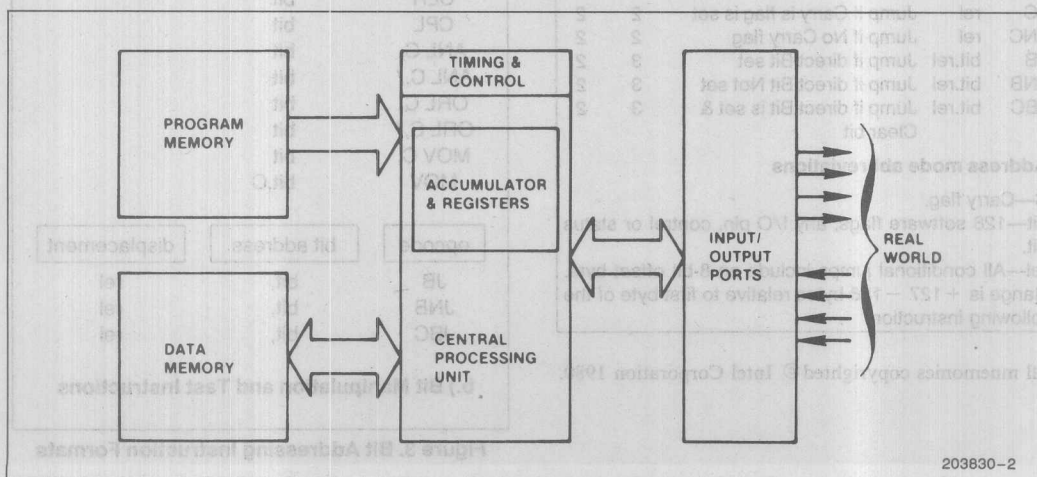


Figure 2. Block Diagram for Abstract Digital Computer

Processing Elements

The introduction stated that the 8051's bit-handling capabilities alone would be sufficient to solve some control applications. Let's see how the four basic elements of a digital computer—a CPU with associated registers, program memory, addressable data RAM, and I/O capability—relate to Boolean variables.

CPU. The 8051 CPU incorporates special logic devoted to executing several bit-wide operations. All told, there are 17 such instructions, all listed in Table 2. Not shown are 94 other (mostly byte-oriented) 8051 instructions.

Program Memory. Bit-processing instructions are fetched from the same program memory as other arithmetic and logical operations. In addition to the instruc-

tions of Table 2, several sophisticated program control features like multiple addressing modes, subroutine nesting, and a two-level interrupt structure are useful in structuring Boolean Processor-based programs.

Boolean instructions are one, two, or three bytes long, depending on what function they perform. Those involving only the carry flag have either a single-byte opcode or an opcode followed by a conditional-branch destination byte (Figure 3a). The more general instructions add a "direct address" byte after the opcode to specify the bit affected, yielding two or three byte encodings (Figure 3b). Though this format allows potentially 256 directly addressable bit locations, not all of them are implemented in the 8051 family.

Table 2. MCS-51™ Boolean Processing Instruction Subset

Mnemonic	Description	Byte	Cyc
SETB C	Set Carry flag	1	1
SETB bit	Set direct Bit	2	1
CLR C	Clear Carry flag	1	1
CLR bit	Clear direct bit	2	1
CPL C	Complement Carry flag	1	1
CPL bit	Complement direct bit	2	1
MOV C,bit	Move direct bit to Carry flag	2	1
MOV bit,C	Move Carry flag to direct bit	2	2
ANL C,bit	AND direct bit to Carry flag	2	2
ANL C,bit	AND complement of direct bit to Carry flag	2	2
ORL C,bit	OR direct bit to Carry flag	2	2
ORL C,bit	OR complement of direct bit to Carry flag	2	2
JC rel	Jump if Carry is flag is set	2	2
JNC rel	Jump if No Carry flag	2	2
JB bit,rel	Jump if direct Bit set	3	2
JNB bit,rel	Jump if direct Bit Not set	3	2
JBC bit,rel	Jump if direct Bit is set & Clear bit	3	2

Address mode abbreviations

C—Carry flag.

bit—128 software flags, any I/O pin, control or status bit.

rel—All conditional jumps include an 8-bit offset byte. Range is +127 -128 bytes relative to first byte of the following instruction.

All mnemonics copyrighted © Intel Corporation 1980.

opcode

SETB C
CLR C
CPL C

opcode

displacement

JC rel
JNC rel

a.) Carry Control and Test Instructions

opcode

bit address

SETB bit
CLR bit
CPL bit
ANL C, bit
ANL C,/ bit
ORL C, bit
ORL C,/ bit
MOV C, bit
MOV bit,C

opcode

bit address

displacement

JB bit, rel
JNB bit, rel
JBC bit, rel

b.) Bit Manipulation and Test Instructions

Figure 3. Bit Addressing Instruction Formats

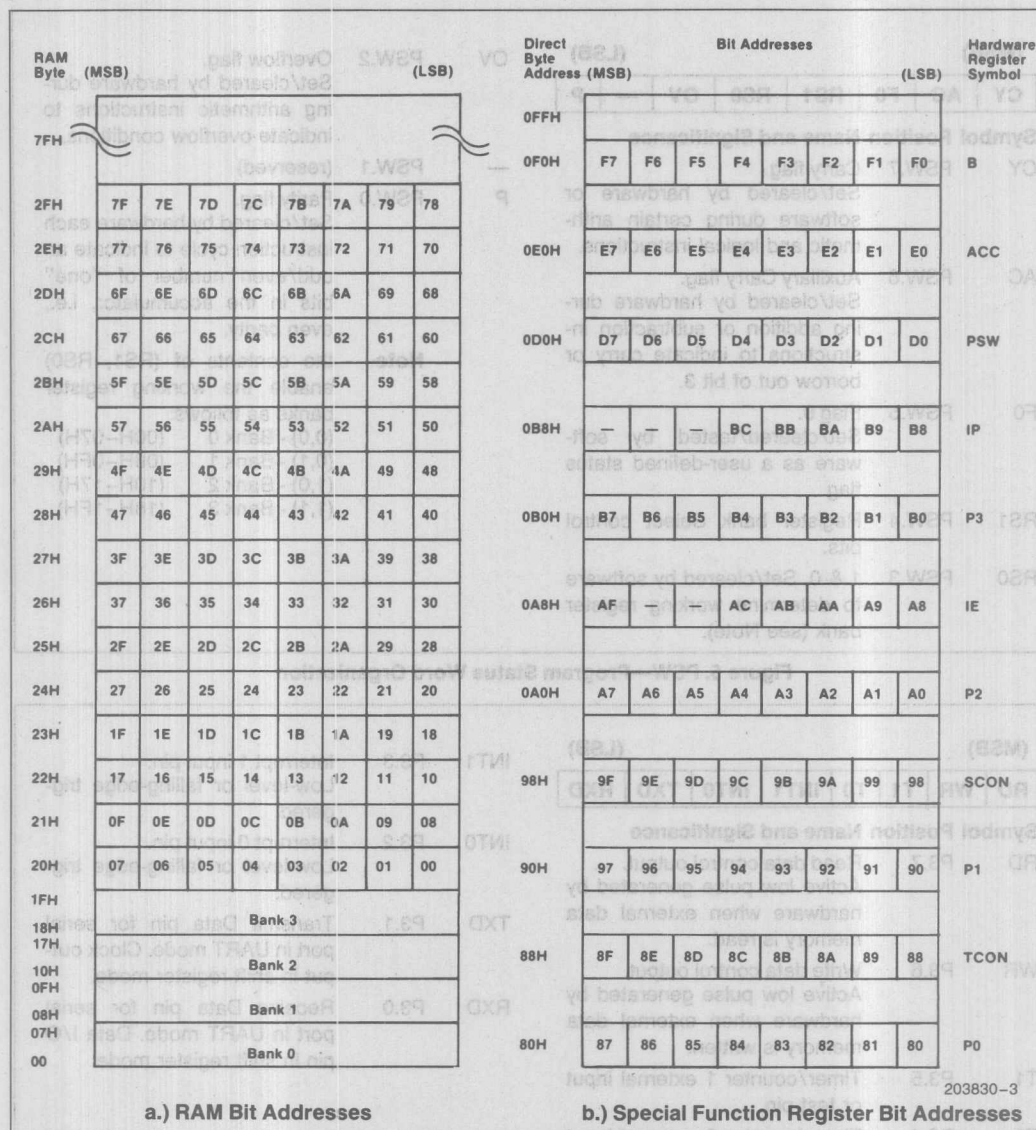


Figure 4. Bit Address Maps

Data Memory. The instructions in Figure 3b can operate directly upon 144 general purpose bits forming the Boolean processor "RAM." These bits can be used as software flags or to store program variables. Two operations and instructions use the CPU's carry flag ("C") as a special one-bit register: in a sense, the carry is a "Boolean accumulator" for logical operations and data transfers.

Input/Output. All 32 I/O pins can be addressed as individual inputs, outputs, or both, in any combination. Any pin can be a control strobe output, status (Test) input, or serial I/O link implemented via software. An additional 33 individually addressable bits reconfigure, control, and monitor the status of the CPU and all on-chip peripheral functions (timer counters, serial port modes, interrupt logic, and so forth).

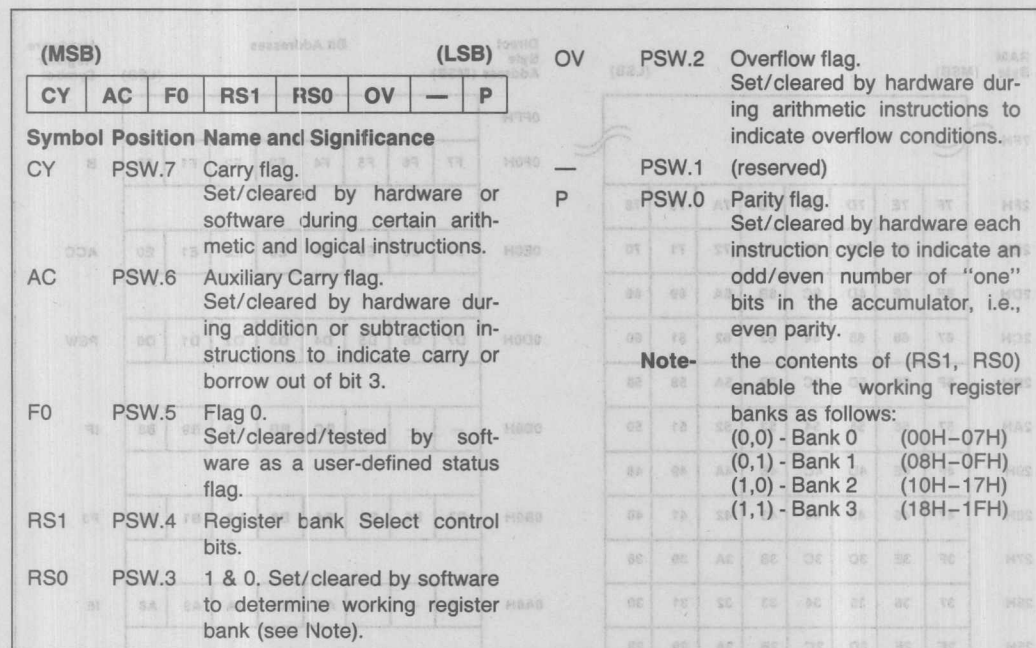


Figure 5. PSW—Program Status Word Organization

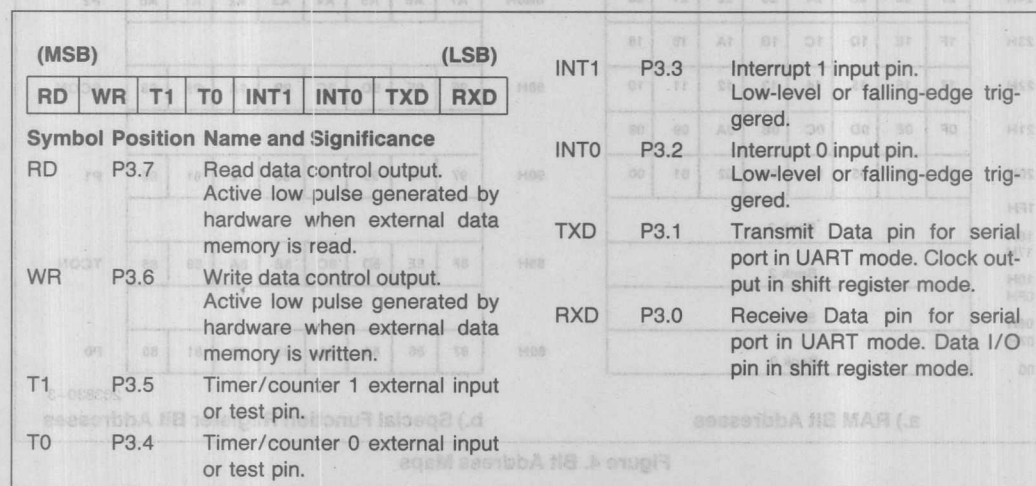


Figure 6. P3—Alternate I/O Functions of Port 3

Direct Bit Addressing

The most significant bit of the direct address byte selects one of two groups of bits. Values between 0 and 127 (00H and 7FH) define bits in a block of 32 bytes of on-chip RAM, between RAM addresses 20H and 2FH (Figure 4a). They are numbered consecutively from the lowest-order byte's lowest-order bit through the highest-order byte's highest-order bit.

Bit addresses between 128 and 255 (80H and 0FFH) correspond to bits in a number of special registers, mostly used for I/O or peripheral control. These positions are numbered with a different scheme than RAM: the five high-order address bits match those of the register's own address, while the three low-order bits identify the bit position within that register (Figure 4b).

Notice the column labeled "Symbol" in Figure 5. Bits with special meanings in the PSW and other registers have corresponding symbolic names. General-purpose (as opposed to carry-specific) instructions may access the carry like any other bit by using the mnemonic CY in place of C. P0, P1, P2, and P3 are the 8051's four I/O ports: secondary functions assigned to each of the eight pins of P3 are shown in Figure 6.

Figure 7 shows the last four bit addressable registers. TCON (Timer Control) and SCON (Serial port Control) control and monitor the corresponding peripherals, while IE (Interrupt Enable) and IP (Interrupt Priority) enable and prioritize the five hardware interrupt sources. Like the reserved hardware register addresses,

the five bits not implemented in IE and IP should not be accessed: they can *not* be used as software flags.

Addressable Register Set. There are 20 special function registers in the 8051, but the advantages of bit addressing only relate to the 11 described below. Five potentially bit-addressable register addresses (0C0H, 0C8H, 0D8H, 0E8H, & 0F8H) are being reserved for possible future expansion in microcomputers based on the MCS-51 architecture. Reading or writing non-existent registers in the 8051 series is pointless, and may cause unpredictable results. Byte-wide logical operations can be used to manipulate bits in all *non-bit* addressable registers and RAM.

a.) TCON—Timer/Counter Control/Status Register		b.) SCON—Serial Port Control/Status Register	
Symbol	Bit	Symbol	Bit
TF0	7	TFB	7
TR0	6	TRB	6
TF1	5	TRF	5
TR1	4	TRB	4
TF2	3	TRF	3
TR2	2	TRB	2
TF3	1	TRF	1
TR3	0	TRB	0
Note: TF0, TR0, TF1, TR1, TF2, TR2, TF3, TR3 are cleared by hardware on timer/counter overflow. TF0, TR0, TF1, TR1, TF2, TR2, TF3, TR3 are set by hardware on timer/counter overflow. TF0, TR0, TF1, TR1, TF2, TR2, TF3, TR3 are cleared by software to turn timer/counter on/off.		Note: TFB, TRB, TRF, TRB are cleared by hardware to de-termining state of ninth data bit transmitted in 8-bit UART mode.	
c.) IE—Interrupt Enable Register		d.) IP—Interrupt Priority Register	
IE	7	IP	7
EA	6	IP	6
EA	5	IP	5
EA	4	IP	4
EA	3	IP	3
EA	2	IP	2
EA	1	IP	1
EA	0	IP	0
Note: IE, EA, EA, EA, EA, EA, EA, EA are cleared by hardware on timer/counter overflow. IE, EA, EA, EA, EA, EA, EA, EA are set by hardware on timer/counter overflow. IE, EA, EA, EA, EA, EA, EA, EA are cleared by software to turn timer/counter on/off.		Note: IP, IP, IP, IP, IP, IP, IP, IP are cleared by hardware to de-termining state of ninth data bit transmitted in 8-bit UART mode.	

Figure 7. Peripheral Configuration Registers

(MSB)				(LSB)			
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Symbol Position Name and Significance							
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.					
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn timer/counter on/off.					
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.					
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn timer/counter on/off.					

a.) TCON—Timer/Counter Control/Status Register

(MSB)				(LSB)			
IE1	IT1	IE0	IT0				
Symbol Position Name and Significance							
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.					
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.					
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.					
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.					

(MSB)								(LSB)							
SM0	SM1	SM2	REN	TB8	RB8	TI	RI								
Symbol Position Name and Significance															
SM0	SCON.7	Serial port Mode control bit 0. Set/cleared by software (see note).													
SM1	SCON.6	Serial port Mode control bit 1. Set/cleared by software (see note).													
SM2	SCON.5	Serial port Mode control bit 2. Set by software to disable reception of frames for which bit 8 is zero.													
REN	SCON.4	Receiver Enable control bit. Set/cleared by software to enable/disable serial data reception.													
TB8	SCON.3	Transmit Bit 8. Set/cleared by hardware to determine state of ninth data bit transmitted in 9-bit UART mode.													

b.) SCON—Serial Port Control/Status Register

RB8	SCON.2	Receive Bit 8. Set/cleared by hardware to indicate state of ninth data bit received.
TI	SCON.1	Transmit Interrupt flag. Set by hardware when byte transmitted. Cleared by software after servicing.
RI	SCON.0	Receive Interrupt flag. Set by hardware when byte received. Cleared by software after servicing.
Note-		the state of (SM0, SM1) selects: (0,0)—Shift register I/O expansion. (0,1)—8-bit UART, variable data rate. (1,0)—9-bit UART, fixed data rate. (1,1)—9-bit UART, variable data rate.

Figure 7. Peripheral Configuration Registers

(MSB)			(LSB)				
EA	—	—	ES	ET1	EX1	ET1	EX0
Symbol Position Name and Significance							
EA	IE.7	Enable All control bit. Cleared by software to disable all interrupts, independent of the state of IE.4–IE.0.					
—	IE.6	(reserved)					
—	IE.5	(reserved)					
ES	IE.4	Enable Serial port control bit. Set/cleared by software to enable/disable interrupts from TI or RI flags.					
ET1	IE.3	Enable Timer 1 control bit. Set/cleared by software to enable/disable interrupts from timer/counter 1.					

c.) IE—Interrupt Enable Register

(MSB)			(LSB)				
—	—	—	PS	PT1	PX1	PT0	PX0
Symbol Position Name and Significance							
—	IP.7	(reserved)					
—	IP.6	(reserved)					
—	IP.5	(reserved)					
PS	IP.4	Serial port Priority control bit. Set/cleared by software to specify high/low priority interrupts for Serial port.					
PT1	IP.3	Timer 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer/counter 1.					

d.) IP—Interrupt Priority Control Register

EX1	IE.2	Enable External interrupt 1 control bit. Set/cleared by software to enable/disable interrupts from INT1.
ET0	IE.1	Enable Timer 0 control bit. Set/cleared by software to enable/disable interrupts from timer/counter 0.
EX0	IE.0	Enable External interrupt 0 control bit. Set/cleared by software to enable/disable interrupts from INTO.

PX1	IP.2	External interrupt 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INT1.
PT0	IP.1	Timer 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer/counter 0.
PX0	IP.0	External interrupt 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INTO.

Figure 7. Peripheral Configuration Registers (Continued)

The accumulator and B registers (A and B) are normally involved in byte-wide arithmetic, but their individual bits can also be used as 16 general software flags. Added with the 128 flags in RAM, this gives 144 general purpose variables for bit-intensive programs. The program status word (PSW) in Figure 5 is a collection of flags and machine status bits including the carry flag itself. Byte operations acting on the PSW can therefore affect the carry.

Instruction Set

Having looked at the bit variables available to the Boolean Processor, we will now look at the four classes of

instructions that manipulate these bits. It may be helpful to refer back to Table 2 while reading this section.

State Control. Addressable bits or flags may be set, cleared, or logically complemented in one instruction cycle with the two-byte instructions SETB, CLR, and CPL. (The "B" affixed to SETB distinguishes it from the assembler "SET" directive used for symbol definition.) SETB and CLR are analogous to loading a bit with a constant: 1 or 0. Single byte versions perform the same three operations on the carry.

The MCS-51 assembly language specifies a bit address in any of three ways:

- by a number or expression corresponding to the direct bit address (0–255):

- by the name or address of the register containing the bit, the *dot operator* symbol (a period: "."), and the bit's position in the register (7-0):
- in the case of control and status registers, by the predefined assembler symbols listed in the first columns of Figures 5-7.

Bits may also be given user-defined names with the assembler "BIT" directive and any of the above techniques. For example, bit 5 of the PSW may be cleared by any of the four instructions.

```

USR_FLG BIT PSW.5 ; User Symbol Definition
.....
CLR OD5H ; Absolute Addressing
CLR PSW.5 ; Use of Dot Operator
CLR FO ; Pre-Defined Assembler
; Symbol
CLR USR_FLG ; User-Defined Symbol
    
```

Data Transfers. The two-byte MOV instructions can transport any addressable bit to the carry in one cycle, or copy the carry to the bit in two cycles. A bit can be moved between two arbitrary locations via the carry by combining the two instructions. (If necessary, push and pop the PSW to preserve the previous contents of the carry.) These instructions can replace the multi-instruction sequence of Figure 8, a program structure appearing in controller applications whenever flags or outputs are conditionally switched on or off.

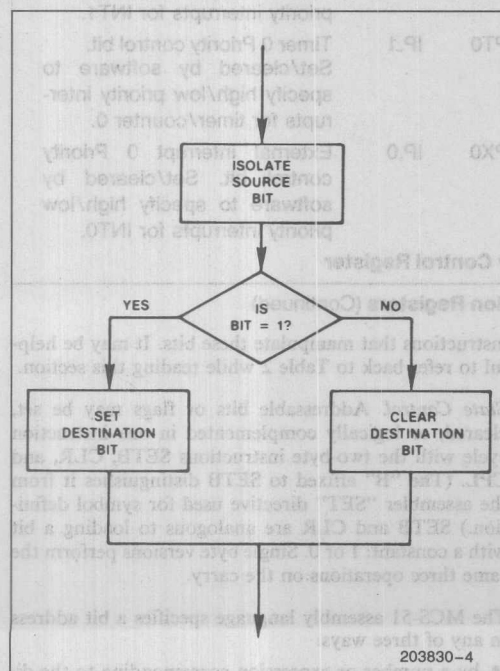


Figure 8. Bit Transfer Instruction Operation

Logical Operations. Four instructions perform the logical-AND and logical-OR operations between the carry and another bit, and leave the results in the carry. The instruction mnemonics are ANL and ORL; the absence or presence of a slash mark ("/") before the source operand indicates whether to use the positive-logic value or the logical complement of the addressed bit. (The source operand itself is never affected.)

Bit-test Instructions. The conditional jump instructions "JC rel" (Jump on Carry) and "JNC rel" (Jump on Not Carry) test the state of the carry flag, branching if it is a one or zero, respectively. (The letters "rel" denote relative code addressing.) The three-byte instructions "JB bit.rel" and "JNB bit.rel" (Jump on Bit and Jump on Not Bit) test the state of *any* addressable bit in a similar manner. A fifth instruction combines the Jump on Bit and Clear operations. "JBC bit.rel" conditionally branches to the indicated address, then clears the bit in the same two cycle instruction. This operation is the same as the MCS-48 "JTF" instructions.

All 8051 conditional jump instructions use program counter-relative addressing, and all execute in two cycles. The last instruction byte encodes a signed displacement ranging from -128 to +127. During execution, the CPU adds this value to the incremented program counter to produce the jump destination. Put another way, a conditional jump to the immediately following instruction would encode 00H in the offset byte.

A section of program or subroutine written using only relative jumps to nearby addresses will have the same machine code independent of the code's location. An assembled routine may be repositioned anywhere in memory, even crossing memory page boundaries, without having to modify the program or recompute destination addresses. To facilitate this flexibility, there is an unconditional "Short Jump" (SJMP) which uses relative addressing as well. Since a programmer would have quite a chore trying to compute relative offset values from one instruction to another, ASM51 automatically computes the displacement needed given only the destination address or label. An error message will alert the programmer if the destination is "out of range."

The so-called "Bit Test" instructions implemented on many other microprocessors simply perform the logical-AND operation between a byte variable and a constant mask, and set or clear a zero flag depending on the result. This is essentially equivalent to the 8051 "MOV C.bit" instruction. A second instruction is then needed to conditionally branch based on the state of the zero flag. This does *not* constitute abstract bit-addressing in the MCS-51 sense. A flag exists only as a field

within a register: to reference a bit the programmer must know and specify both the encompassing register and the bit's position therein. This constraint severely limits the flexibility of symbolic bit addressing and reduces the machine's code-efficiency and speed.

Interaction with Other Instructions. The carry flag is also affected by the instructions listed in Table 3. It can be rotated through the accumulator, and altered as a side effect of arithmetic instructions. Refer to the User's Manual for details on how these instructions operate.

Simple Instruction Combinations

By combining general purpose bit operations with certain addressable bits, one can "custom build" several hundred useful instructions. All eight bits of the PSW can be tested directly with conditional jump instructions to monitor (among other things) parity and overflow status. Programmers can take advantage of 128 software flags to keep track of operating modes, resource usage, and so forth.

The Boolean instructions are also the most efficient way to control or reconfigure peripheral and I/O registers. All 32 I/O lines become "test pins," for example, tested by conditional jump instructions. Any output pin can be toggled (complemented) in a single instruction cycle. Setting or clearing the Timer Run flags (TR0 and TR1) turn the timer/counters on or off; polling the same flags elsewhere lets the program determine if a timer is running. The respective overflow flags (TF0 and TF1) can be tested to determine when the desired period or count has elapsed, then cleared in preparation for the next repetition. (For the record, these bits are all part of the TCON register, Figure 7a. Thanks to symbolic bit addressing, the programmer only needs to remember the mnemonic associated with each function. In other words, don't bother memorizing control word layouts.)

In the MCS-48 family, instructions corresponding to some of the above functions require specific opcodes. Ten different opcodes serve to clear/complement the software flags F0 and F1, enable/disable each interrupt, and start/stop the timer. In the 8051 instruction set, just three opcodes (SETB, CLR, CPL) with a direct bit address appended perform the same functions. Two test instructions (JB and JNB) can be combined with bit addresses to test the software flags, the 8048 I/O pins T0, T1, and INT, and the eight accumulator bits, replacing 15 more different instructions.

Table 4a shows how 8051 programs implement software flag and machine control functions associated with special opcodes in the 8048. In every case the MCS-51 solution requires the same number of machine cycles, and executes 2.5 times faster.

Table 3. Other Instructions Affecting the Carry Flag

Mnemonic	Description	Byte	Cyc
ADD A,Rn	Add register to Accumulator	1	1
ADD A,direct	Add direct byte to Accumulator	2	1
ADD A,@Ri	Add indirect RAM to Accumulator	1	1
ADD A,#data	Add immediate data to Accumulator	2	1
ADDC A,Rn	Add register to Accumulator with Carry flag	1	1
ADDC A,direct	Add direct byte to Accumulator with Carry flag	2	1
ADDC A,@Ri	Add indirect RAM to Accumulator with Carry flag	1	1
ADDC A,#data	Add immediate data to Acc with Carry flag	2	1
SUBB A,Rn	Subtract register from Accumulator with borrow	1	1
SUBB A,direct	Subtract direct byte from Acc with borrow	2	1
SUBB A,@Ri	Subtract indirect RAM from Acc with borrow	1	1
SUBB A,#data	Subtract immediate data from Acc with borrow	2	1
MUL AB	Multiply A & B	1	4
DIV AB	Divide A by B	1	4
DA A	Decimal Adjust Accumulator	1	1
RLC A	Rotate Accumulator Left through the Carry flag	1	1
RRC A	Rotate Accumulator Right through Carry flag	1	1
CJNE A,direct,rel	Compare direct byte to Acc & Jump if Not Equal	3	2
CJNE A,#data,rel	Compare immediate to Acc & Jump if Not Equal	3	2
CJNE Rn,#data,rel	Compare immed to register & Jump if Not Equal	3	2
CJNE @Ri,#data,rel	Compare immed to indirect & Jump if Not Equal	3	2

All mnemonics copyrighted © Intel Corporation 1980.

Table 4a. Contrasting 8048 and 8051 Bit Control and Testing Instructions

8048 Instruction	Bytes	Cycles	μSec	8x51 Instruction	Bytes	Cycles & μSec
Flag Control						
CLR C	1	1	2.5	CLR C	1	1
CPL F0	1	1	2.5	CPL F0	2	1
Flag Testing						
JNC offset	2	2	5.0	JNC rel	2	2
JF0 offset	2	2	5.0	JB F0.rel	3	2
JB7 offset	2	2	5.0	JB ACC.7.rel	3	2
Peripheral Polling						
JT0 offset	2	2	5.0	JB T0.rel	3	2
JN1 offset	2	2	5.0	JNB INT0.rel	3	2
JTF offset	2	2	5.0	JBC TF0.rel	3	2
Machine and Peripheral Control						
STRT T	1	1	2.5	SETB TR0	2	1
EN 1	1	1	2.5	SETB EX0	2	1
DIS TCNT1	1	1	2.5	CLR ET0	2	1

Table 4b. Replacing 8048 Instruction Sequences with Single 8x51 Instructions

8048 Instruction	Bytes	Cycles	μSec	8051 Instruction	Bytes	Cycles & μSec
Flag Control						
Set carry						
CLR C	1	1	2.5	SETB C	1	1
CPL F0	1	1	2.5			
Set Software Flag						
CLR F0	1	1	2.5	SETB F0	2	1
CPL F0	1	1	2.5			
Turn Off Output Pin						
ANL P1.#0FBH	2	2	5.0	CLR P1.2	2	1
Complement Output Pin						
IN A.P1	1	1	2.5			
XRL A.#04H	1	1	2.5			
OUTL P1.A	4	6	15.0	CPL P1.2	2	1
Clear Flag in RAM						
MOV R0.#FLGADR	1	1	2.5			
MOV A.@R0	1	1	2.5			
ANL A.#FLGMASK	1	1	2.5			
MOV @R0.A	6	6	15.0	CLR USER_FLG	2	1

Table 4b. Replacing 8048 Instruction Sequences with Single 8x51 Instructions (Continued)

8048 Instruction	Bytes	Cycles	μSec	8x51 Instruction	Bytes	Cycles & μSec
Flag Testing:						
Jump if Software Flag is 0						
JFO \$+4				JNB F0.rel	3	2
JMP offset = 4	4	4	10.0			
Jump if Accumulator bit is 0						
CPL A				JNB ACC.7.rel	3	2
JB7 offset						
CPL A = 4	4	4	10.0			
Peripheral Polling						
Test if Input Pin is Grounded						
IN A.P1				JNB P1.3.rel	3	2
CPL A						
JB3 offset = 4	4	5	12.5			
Test if Interrupt Pin is High						
JN1 \$+4				JB INT0.rel	3	2
JMP offset = 4	4	4	10.0			

3.0 BOOLEAN PROCESSOR APPLICATIONS

So what? Then what does all this buy you?

Qualitatively, nothing. All the same capabilities could be (and often have been) implemented on other machines using awkward sequences of other basic operations. As mentioned earlier, any CPU can solve any problem given enough time.

Quantitatively, the differences between a solution allowed by the 8051 and those required by previous architectures are numerous. What the 8051 Family buys you is a faster, cleaner, lower-cost solution to micro-controller applications.

The opcode space freed by condensing many specific 8048 instructions into a few general operations has been used to add new functionality to the MCS-51 architecture—both for byte and bit operations. 144 software flags replace the 8048's two. These flags (and the carry) may be directly set, not just cleared and complemented, and all can be tested for either state, not just one. Operating mode bits previously inaccessible may be read, tested, or saved. Situations where the 8051 instruction set provides new capabilities are contrasted with 8048 instruction sequences in Table 4b. Here the 8051 speed advantage ranges from 5x to 15x!

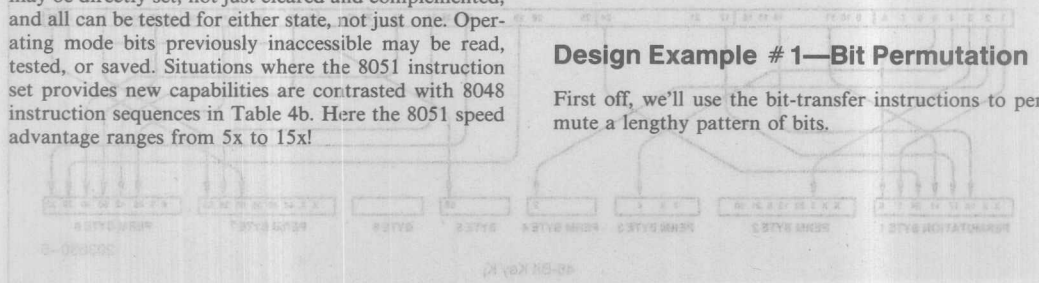
Combining Boolean and byte-wide instructions can produce great synergy. An MCS-51 based application will prove to be:

- simpler to write since the architecture correlates more closely with the problems being solved;
- easier to debug because more individual instructions have no unexpected or undesirable side-effects;
- more byte efficient due to direct bit addressing and program counter relative branching;
- faster running because fewer bytes of instruction need to be fetched and fewer conditional jumps are processed;
- lower cost because of the high level of system-integration within one component.

These rather unabashed claims of excellence shall not go unsubstantiated. The rest of this chapter examines less trivial tasks simplified by the Boolean processor. The first three compare the 8051 with other micro-processors; the last two go into 8051-based system designs in much greater depth.

Design Example #1—Bit Permutation

First off, we'll use the bit-transfer instructions to permute a lengthy pattern of bits.



A steadily increasing number of data communication products use encoding methods to protect the security of sensitive information. By law, interstate financial transactions involving the Federal banking system must be transmitted using the Federal Information Processing *Data Encryption Standard* (DES).

Basically, the DES combines eight bytes of "plaintext" data (in binary, ASCII, or any other format) with a 56-bit "key", producing a 64-bit encrypted value for transmission. At the receiving end the same algorithm is applied to the incoming data using the same key, reproducing the original eight byte message. The algorithm used for these permutations is fixed; different user-defined keys ensure data privacy.

It is not the purpose of this note to describe the DES in any detail. Suffice it to say that encryption/decryption is a long, iterative process consisting of rotations, exclusive -OR operations, function table look-ups, and an extensive (and quite bizarre) sequence of bit permutation, packing, and unpacking steps. (For further details refer to the June 21, 1979 issue of *Electronics* magazine.) The bit manipulation steps are included, it is rumored, to impede a general purpose digital supercomputer trying to "break" the code. Any algorithm implementing the DES with previous generation microprocessors would spend virtually all of its time diddling bits.

The bit manipulation performed is typified by the Key Schedule Calculation represented in Figure 9. This step is repeated 16 times for each key used in the course of a transmission. In essence, a seven-byte, 56-bit "Shifted Key Buffer" is transformed into an eight-byte, "Permutation Buffer" without altering the shifted Key. The arrows in Figure 9 indicate a few of the translation steps. Only six bits of each byte of the Permutation Buffer are used; the two high-order bits of each byte are cleared. This means only 48 of the 56 Shifted Key Buffer bits are used in any one iteration.

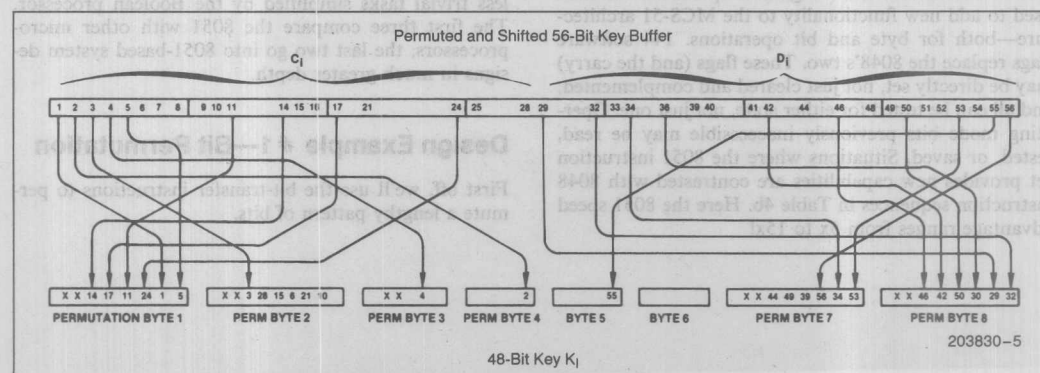


Figure 9. DES Key Schedule Transformation

Different microprocessor architectures would best implement this type of permutation in different ways. Most approaches would share the steps of Figure 10a:

- Initialize the Permutation Buffer to default state (ones or zeroes):
- Isolate the state of a bit of a byte from the Key Buffer. Depending on the CPU, this might be accomplished by rotating a word of the Key Buffer through a carry flag or testing a bit in memory or an accumulator against a mask byte:
- Perform a conditional jump based on the carry or zero flag if the Permutation Buffer default state is correct:
- Otherwise reverse the corresponding bit in the permutation buffer with logical operations and mask bytes.

Each step above may require several instructions. The last three steps must be repeated for all 48 bits. Most microprocessors would spend 300 to 3,000 microseconds on each of the 16 iterations.

Notice, though, that this flow chart looks a lot like Figure 8. The Boolean Processor can permute bits by simply moving them from the source to the carry to the destination—a total of two instructions taking four bytes and three microseconds per bit. Assume the Shifted Key Buffer and Permutation Buffer both reside in bit-addressable RAM, with the bits of the former assigned symbolic names SKB_1, SKB_2, ... SKB_56, and that the bytes of the latter are named PB_1, ... PB_8. Then working from Figure 9, the software for the permutation algorithm would be that of Example 1a. The total routine length would be 192 bytes, requiring 144 microseconds.

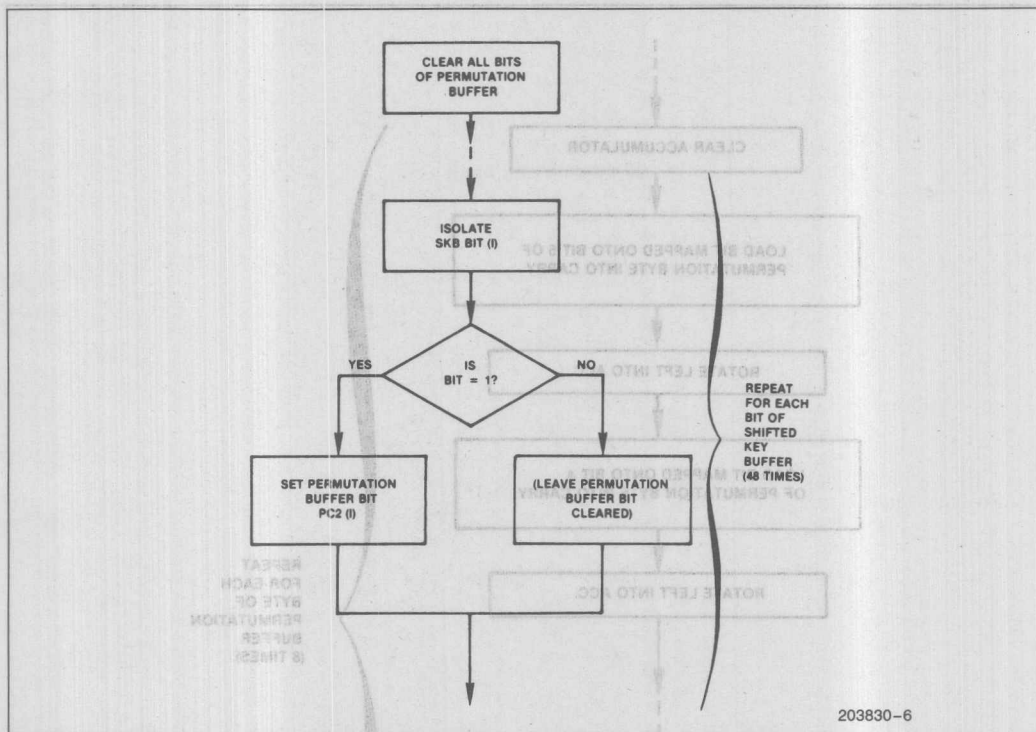


Figure 10a. Flowchart for Key Permutation Attempted with a Byte Processor

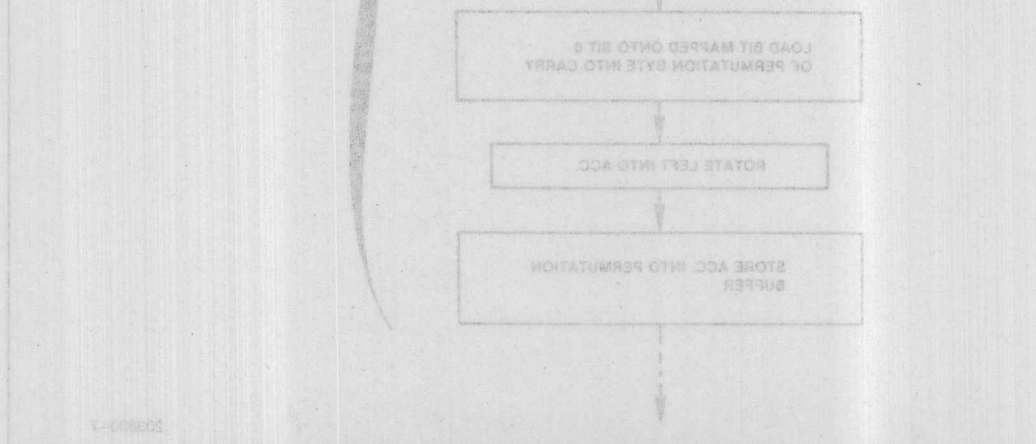


Figure 10b. DES Key Permutation with Boolean Processor

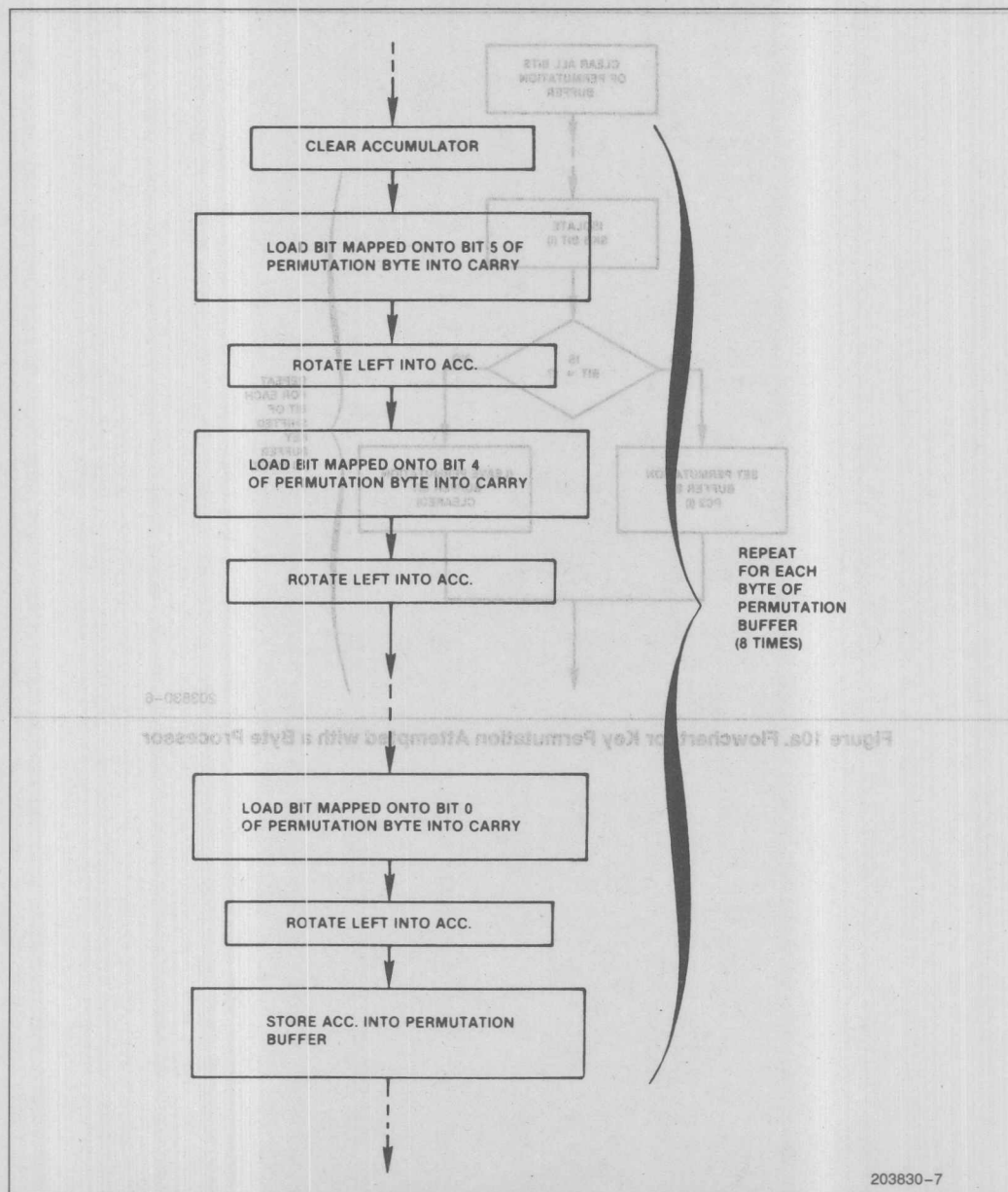


Figure 10b. DES Key Permution with Boolean Processor

The algorithm of Figure 10b is just slightly more efficient in this time-critical application and illustrates the synergy of an integrated byte and bit processor. The bits needed for each byte of the Permutation Buffer are assimilated by loading each bit into the carry (1 μ s.) and shifting it into the accumulator (1 μ s.). Each byte is stored in RAM when completed. Forty-eight bits thus need a total of 112 instructions, some of which are listed in Example 1b.

Worst-case execution time would be 112 microseconds, since each instruction takes a single cycle. Routine length would also decrease, to 168 bytes. (Actually, in the context of the complete encryption algorithm, each permuted byte would be processed as soon as it is assimilated—saving memory and cutting execution time by another 8 μ s.)

To date, most banking terminals and other systems using the DES have needed special boards or peripheral controller chips just for the encryption/decryption process, and still more hardware to form a serial bit stream for transmission (Figure 11a). An 8051 solution could pack most of the entire system onto the one chip (Figure 11b). The whole DES algorithm would require less than one-fourth of the on-chip program memory, with the remaining bytes free for operating the banking terminal (or whatever) itself.

Moreover, since transmission and reception of data is performed through the on-board UART, the unencrypted data (plaintext) never even exists outside the microcomputer! Naturally, this would afford a high degree of security from data interception.

Example 1. DES Key Permutation Software.

a.) "Brute Force" technique

```
MOV C,SKB_1
MOV PB_1.1,C
MOV C,SKB_2
MOV PB_4.0,C
MOV C,SKB_3
MOV PB_2.5,C
MOV C,SKB_4
MOV PB_1.0,C
...
...
MOV C,SKB_55
MOV PB_5.0,C
MOV C,SKB_56
MOV PB_7.2,C
```

b.) Using Accumulator to Collect Bits

```
CLR A
MOV C,SKB_14
RLC A
MOV C,SKB_17
RLC A
MOV C,SKB_11
RLC A
MOV C,SKB_24
RLC A
MOV C,SKB_1
RLC A
MOV C,SKB_5
RLC A
MOV PB_1,A
...
...
MOV C,SKB_29
RLC A
MOV C,SKB_32
RLC A
MOV PB_8,A
```

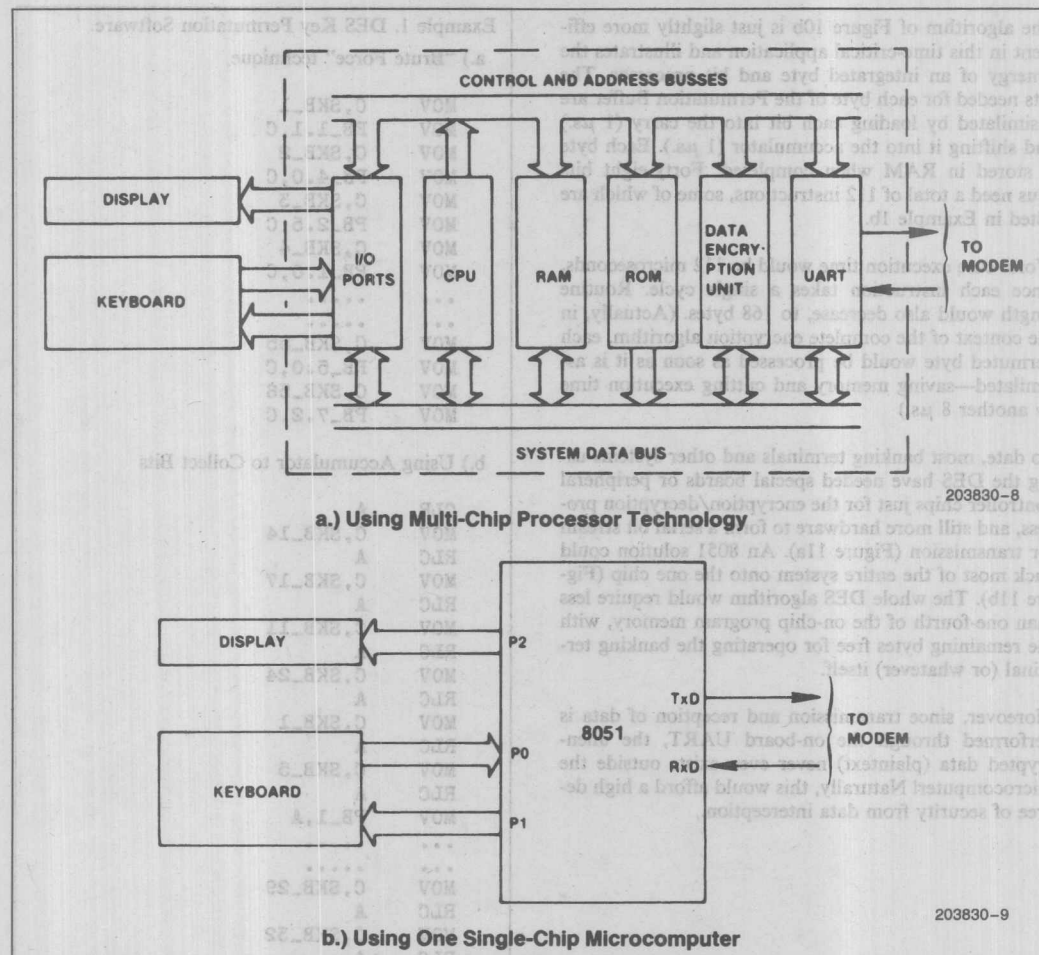


Figure 11. Secure Banking Terminal Block Diagram

Design Example #2—Software Serial I/O

An exercise often imposed on beginning microcomputer students is to write a program simulating a UART. Though doing this with the 8051 Family may appear to be a moot point (given that the hardware for a full UART is on-chip), it is still instructive to see how it would be done, and maintains a product line tradition.

As it turns out, the 8051 microcomputers can receive or transmit serial data via software very efficiently using the Boolean instruction set. Since any I/O pin may be a serial input or output, several serial links could be maintained at once.

Figures 12a and 12b show algorithms for receiving or transmitting a byte of data. (Another section of program would invoke this algorithm eight times, synchronizing it with a start bit, clock signal, software delay, or timer interrupt.) Data is received by testing an input pin, setting the carry to the same state, shifting the carry into a data buffer, and saving the partial frame in internal RAM. Data is transmitted by shifting an output buffer through the carry, and generating each bit on an output pin.

A side-by-side comparison of the software for this common "bit-banging" application with three different microprocessor architectures is shown in Table 5a and 5b. The 8051 solution is more efficient than the others on every count!

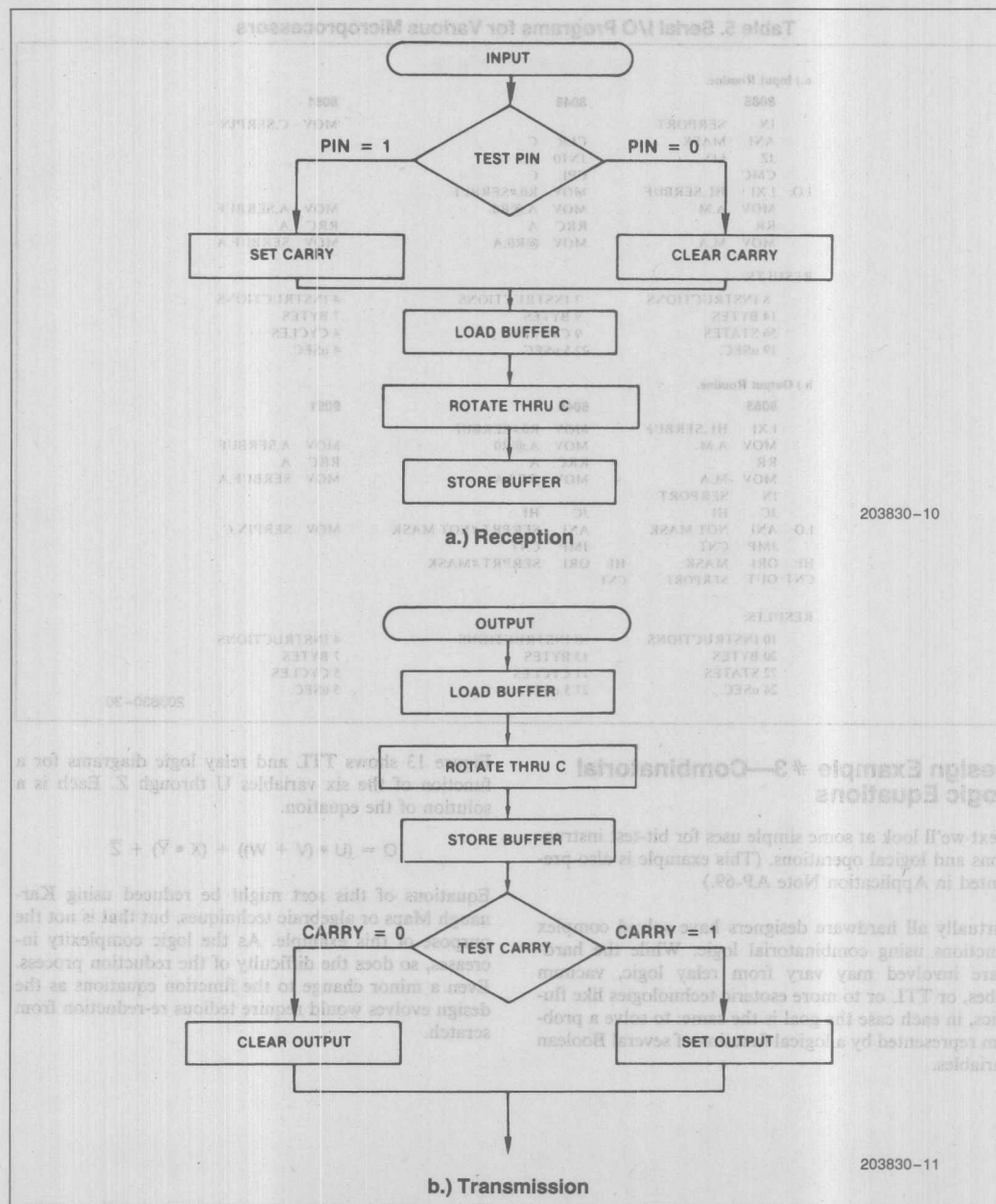


Figure 12. Serial I/O Algorithms

Table 5. Serial I/O Programs for Various Microprocessors

a.) Input Routine.		
8085	8048	8051
IN SERPORT ANI MASK JZ LO CMC I.O: LXI HL, SERBUF MOV A, M RR MOV M, A	CLR C JNT0 I.O TEST CPI C MOV R0, #SERBUF MOV A, @R0 RRC A MOV @R0, A	MOV C, SERPIN MOV A, SERBUF RRC A MOV SERBUF, A
RESULTS: 8 INSTRUCTIONS 14 BYTES 56 STATES 19 uSEC.	7 INSTRUCTIONS 9 BYTES 9 CYCLES 22.5 uSEC.	4 INSTRUCTIONS 7 BYTES 4 CYCLES 4 uSEC.
b.) Output Routine.		
8085	8048	8051
LXI HL, SERBUF MOV A, M RR MOV M, A IN SERPORT JC HI I.O: ANI NOT MASK JMP CNT HI: ORI MASK CNT: OUT SERPORT	MOV R0, #SERBUF MOV A, @R0 RRC A MOV @R0, A JC HI ANI SERPRT, #NOT MASK JMP CNT HI: ORI SERPRT, #MASK CNT:	MOV A, SERBUF RRC A MOV SERBUF, A MOV SERPIN, C
RESULTS: 10 INSTRUCTIONS 20 BYTES 72 STATES 24 uSEC.	8 INSTRUCTIONS 13 BYTES 11 CYCLES 27.5 uSEC.	4 INSTRUCTIONS 7 BYTES 5 CYCLES 5 uSEC.

203830-30

Design Example #3—Combinatorial Logic Equations

Next we'll look at some simple uses for bit-test instructions and logical operations. (This example is also presented in Application Note AP-69.)

Virtually all hardware designers have solved complex functions using combinatorial logic. While the hardware involved may vary from relay logic, vacuum tubes, or TTL or to more esoteric technologies like fluidics, in each case the goal is the same: to solve a problem represented by a logical function of several Boolean variables.

Figure 13 shows TTL and relay logic diagrams for a function of the six variables U through Z. Each is a solution of the equation.

$$Q = (U \cdot (V + W)) + (X \cdot \bar{Y}) + \bar{Z}$$

Equations of this sort might be reduced using Karnaugh Maps or algebraic techniques, but that is not the purpose of this example. As the logic complexity increases, so does the difficulty of the reduction process. Even a minor change to the function equations as the design evolves would require tedious re-reduction from scratch.

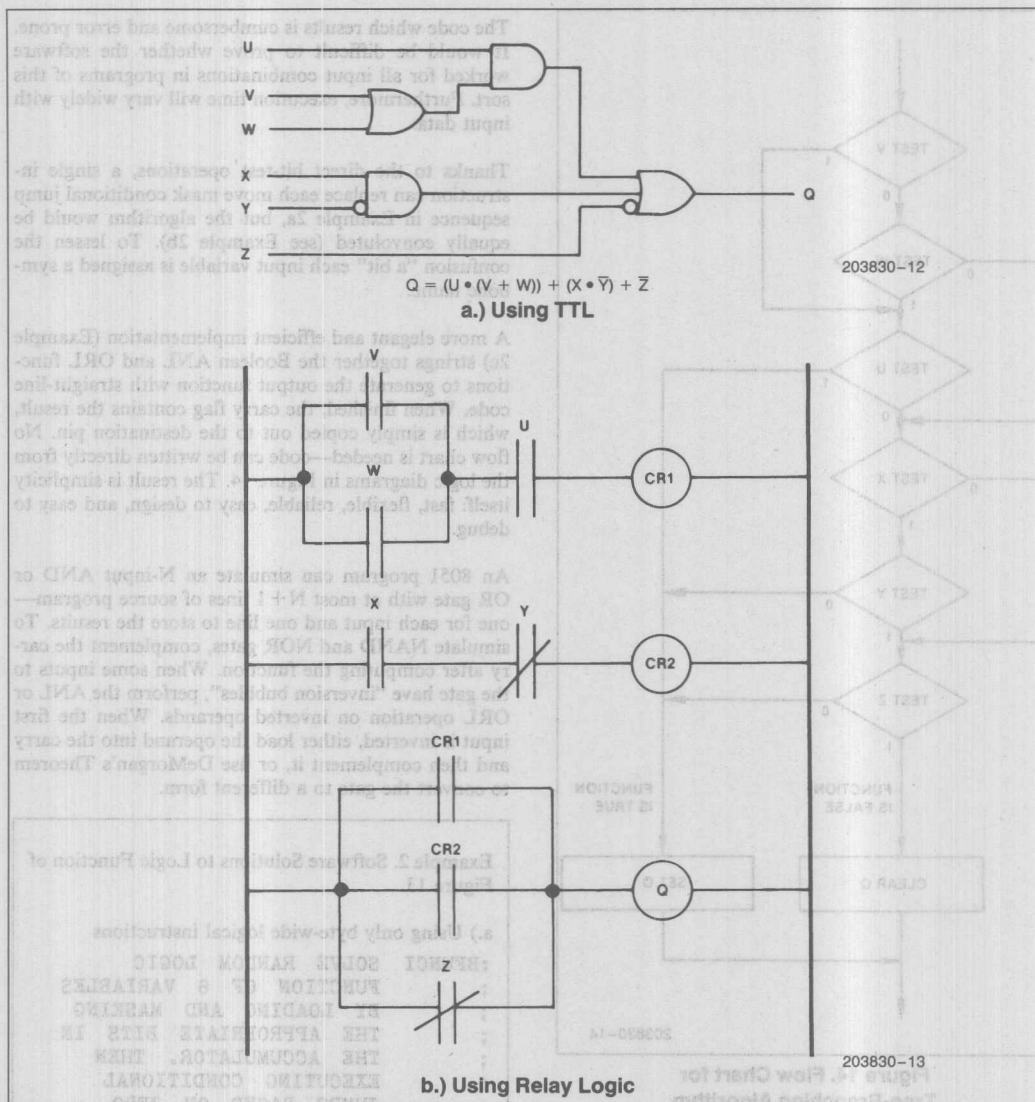


Figure 13. Hardware Implementations of Boolean Functions

For the sake of comparison we will implement this function three ways, restricting the software to three proper subsets of the MCS-51 instruction set. We will also assume that U and V are input pins from different input ports, W and X are status bits for two peripheral controllers, and Y and Z are software flags set up earlier in the program. The end result must be written

to an output pin on some third port. The first two implementations follow the flow-chart shown in Figure 14. Program flow would embark on a route down a test-and-branch tree and leaves either the "True" or "Not True" exit ASAP—as soon as the proper result has been determined. These exits then rewrite the output port with the result bit respectively one or zero.

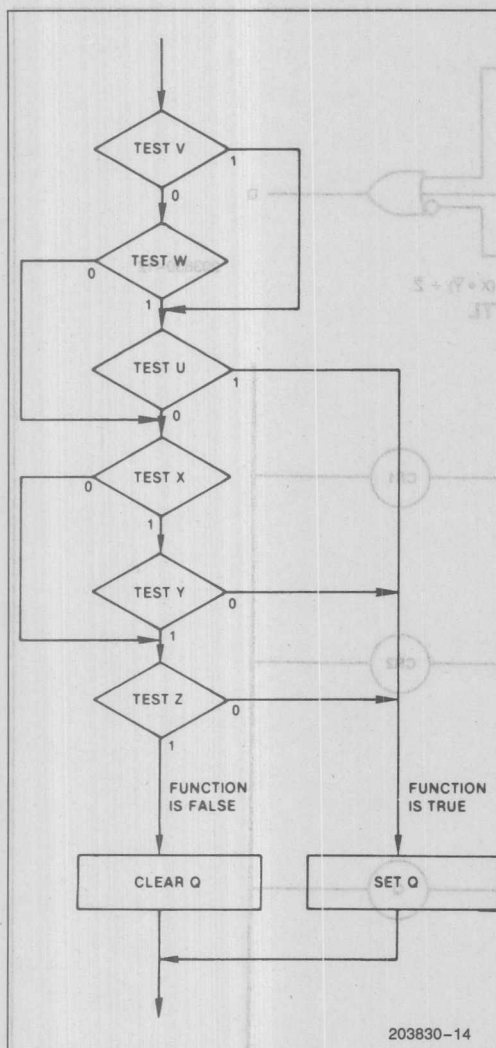


Figure 14. Flow Chart for Tree-Branching Algorithm

Other digital computers must solve equations of this type with standard word-wide logical instructions and conditional jumps. So for the first implementation, we won't use any generalized bit-addressing instructions. As we shall soon see, being constrained to such an instruction subset produces somewhat sloppy software solutions. MCS-51 mnemonics are used in Example 2a; other machines might further cloud the situation by requiring operation-specific mnemonics like INPUT, OUTPUT, LOAD, STORE, etc., instead of the MOV mnemonic used for all variable transfers in the 8051 instruction set.

The code which results is cumbersome and error prone. It would be difficult to prove whether the software worked for all input combinations in programs of this sort. Furthermore, execution time will vary widely with input data.

Thanks to the direct bit-test operations, a single instruction can replace each move mask conditional jump sequence in Example 2a, but the algorithm would be equally convoluted (see Example 2b). To lessen the confusion "a bit" each input variable is assigned a symbolic name.

A more elegant and efficient implementation (Example 2c) strings together the Boolean ANL and ORL functions to generate the output function with straight-line code. When finished, the carry flag contains the result, which is simply copied out to the destination pin. No flow chart is needed—code can be written directly from the logic diagrams in Figure 14. The result is simplicity itself: fast, flexible, reliable, easy to design, and easy to debug.

An 8051 program can simulate an N-input AND or OR gate with at most $N + 1$ lines of source program—one for each input and one line to store the results. To simulate NAND and NOR gates, complement the carry after computing the function. When some inputs to the gate have "inversion bubbles", perform the ANL or ORL operation on inverted operands. When the first input is inverted, either load the operand into the carry and then complement it, or use DeMorgan's Theorem to convert the gate to a different form.

Example 2. Software Solutions to Logic Function of Figure 13.

a.) Using only byte-wide logical instructions

```

:BFUNCI SOLVE RANDOM LOGIC
;      FUNCTION OF 6 VARIABLES
;      BY LOADING AND MASKING
;      THE APPROPRIATE BITS IN
;      THE ACCUMULATOR. THEN
;      EXECUTING CONDITIONAL
;      JUMPS BASED ON ZERO
;      CONDITION. (APPROACH USED
;      BY BYTE-ORIENTED
;      ARCHITECTURES.) BYTE AND
;      MASK VALUES CORRESPOND TO
;      RESPECTIVE BYTE ADDRESS
;      AND BIT POSITIONS.
;
;      OUTBUF DATA 22H
;      OUTPUT PIN STATE MAP
;

```

```

TESTV: MOV A,P2
      ANL A,#00000100B
      JNZ TESTU
      MOV A,TCON
      ANL A,#00100000B
      JZ TESTX
TESTU: MOV A,P1
      ANL A,#00000010B
      JNZ SETQ
TESTX: MOV A,TCON
      ANL A,#00001000B
      JZ TESTZ
      MOV A,20H
      ANL A,#00000001B
      JZ SETQ
TESTZ: MOV A,21H
      ANL A,#00000010B
      JZ SETQ
CLRQ:  MOV A,OUTBUF
      ANL A,#11110111B
      JMP OUTQ
SETQ:  MOV A,OUTBUF
      ORL A,#00001000B
OUTQ:  MOV OUTBUF,A
      MOV P3,A

```

b.) Using only bit-test instructions

```

:BFUNC2 SOLVE A RANDOM LOGIC
; FUNCTION OF 6 VARIABLES
; BY DIRECTLY POLLING EACH
; BIT. (APPROACH USING
; MCS-51 UNIQUE BIT-TEST
; INSTRUCTION CAPABILITY.)
; SYMBOLS USED IN LOGIC
; DIAGRAM ASSIGNED TO
; CORRESPONDING 8x51 BIT
; ADDRESSES.
;
;
;

```

```

U      BIT P1.1
V      BIT P2.2
W      BIT TFO
X      BIT IEL
Y      BIT 20H.0
Z      BIT 21H.1
Q      BIT P3.3
;      ...
TEST_V: JB V,TEST_U
      JNB W,TEST_X
TEST_U: JB U,SET_Q
TEST_X: JNB X,TEST_Z
      JNB Y,SET_Q
TEST_Z: JNB Z,SET_Q
CLR_Q:  CLR Q
      JMP NXTTST
SET_Q:  SETB Q
NXTTST: (CONTINUATION OF
:PROGRAM)

```

c.) Using logical operations on Boolean variables

```

:FUNC3 SOLVE A RANDOM LOGIC
; FUNCTION OF 6 VARIABLES
; USING STRAIGHT_LINE
; LOGICAL INSTRUCTIONS ON
; MCS-51 BOOLEAN VARIABLES.
;
MOV C,V
ORL C,W ;OUTPUT OF OR GATE
ANL C,U ;OUTPUT OF TOP AND GATE
MOV FO,C ;SAVE INTERMEDIATE STATE
MOV C,X
ANL C,Y ;OUTPUT OF BOTTOM AND GATE
ORL C,FO ;INCLUDE VALUE SAVED ABOVE
ORL C,Z ;INCLUDE LAST INPUT
;VARIABLE
MOV Q,C ;OUTPUT COMPUTED RESULT

```


An upper-limit can be placed on the complexity of software to simulate a large number of gates by summing the total number of inputs and outputs. The *actual* total should be somewhat shorter, since calculations can be "chained," as shown. The output of one gate is often the first input to another, bypassing the intermediate variable to eliminate two lines of source.

Design Example #4—Automotive Dashboard Functions

Now let's apply these techniques to designing the software for a complete controller system. This application is patterned after a familiar real-world application which isn't nearly as trivial as it might first appear: automobile turn signals.

Imagine the three position turn lever on the steering column as a single-pole, triple-throw toggle switch. In its central position all contacts are open. In the up or down positions contacts close causing corresponding lights in the rear of the car to blink. So far very simple.

Two more turn signals blink in the front of the car, and two others in the dashboard. All six bulbs flash when an emergency switch is closed. A thermo-mechanical relay (accessible under the dashboard in case it wears out) causes the blinking.

Applying the brake pedal turns the tail light filaments on constantly . . . unless a turn is in progress, in which case the blinking tail light is not affected. (Of course, the front turn signals and dashboard indicators are not affected by the brake pedal.) Table 6 summarizes these operating modes.

Table 6. Truth Table for Turn-Signal Operation

Input Signals				Output Signals			
Brake Switch	Emerg. Switch	Left Turn Switch	Right Turn Switch	Left Front & Dash	Right Front & Dash	Left Rear	Right Rear
0	0	0	0	Off	Off	Off	Off
0	0	0	1	Off	Blink	Off	Blink
0	0	1	0	Blink	Off	Blink	Off
0	1	0	0	Blink	Blink	Blink	Blink
0	1	0	1	Blink	Blink	Blink	Blink
0	1	1	0	Blink	Blink	Blink	Blink
1	0	0	0	Off	Off	On	On
1	0	0	1	Off	Blink	On	Blink
1	0	1	0	Blink	Off	Blink	On
1	1	0	0	Blink	Blink	On	On
1	1	0	1	Blink	Blink	On	Blink
1	1	1	0	Blink	Blink	Blink	On

(but not the dashboard) bulbs has a second, somewhat dimmer filament for the parking lights. Figure 15 shows TTL circuitry which could control all six bulbs. The signals labeled "High Freq." and "Low Freq." represent two square-wave inputs. Basically, when one of the turn switches is closed or the emergency switch is activated the low frequency signal (about 1 Hz) is gated through to the appropriate dashboard indicator(s) and turn signal(s). The rear signals are also activated when the brake pedal is depressed provided a turn is not being made in the same direction. When the parking light switch is closed the higher frequency oscillator is gated to each front and rear turn signal, sustaining a low-intensity background level. (This is to eliminate the need for additional parking light filaments.)

In most cars, the switching logic to generate these functions requires a number of multiple-throw contacts. As many as 18 conductors thread the steering column of some automobiles solely for turn-signal and emergency blinker functions. (The author discovered this recently to his astonishment and dismay when replacing the whole assembly because of one burned contact.)

A multiple-conductor wiring harness runs to each corner of the car, behind the dash, up the steering column, and down to the blinker relay below. Connectors at

and labor during construction, lower reliability and safety, and more costly repairs. And considering the system's present complexity, increasing its reliability or detecting failures would be quite difficult.

There are two reasons for going into such painful detail describing this example. First, to show that the messiest part of many system designs is determining what the controller should do. Writing the software to solve these functions will be comparatively easy. Secondly, to show the many potential failure points in the system. Later we'll see how the peripheral functions and intelligence built into a microcomputer (with a little creativity) can greatly reduce external interconnections and mechanical part count.

The Single-Chip Solution

The circuit shown in Figure 16 indicates five input pins to the five input variables—left-turn select, right-turn select, brake pedal down, emergency switch on, and parking lights on. Six output pins turn on the front, rear, and dashboard indicators for each side. The microcomputer implements all logical functions through software, which periodically updates the output signals as time elapses and input conditions change.

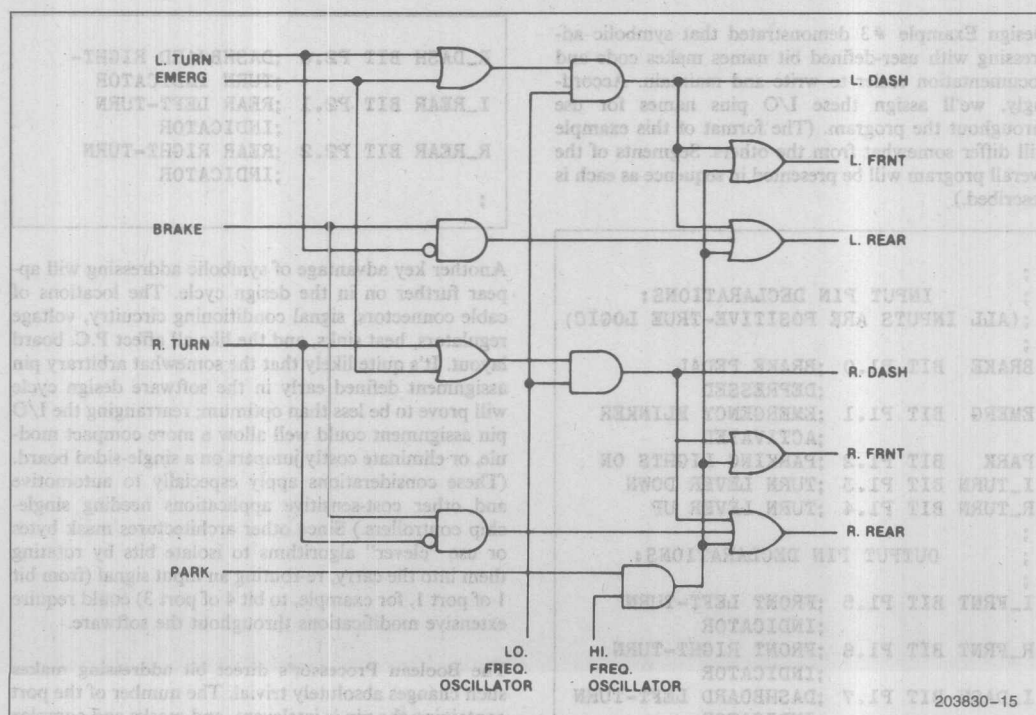


Figure 15. TTL Logic Implementation of Automotive Turn Signals

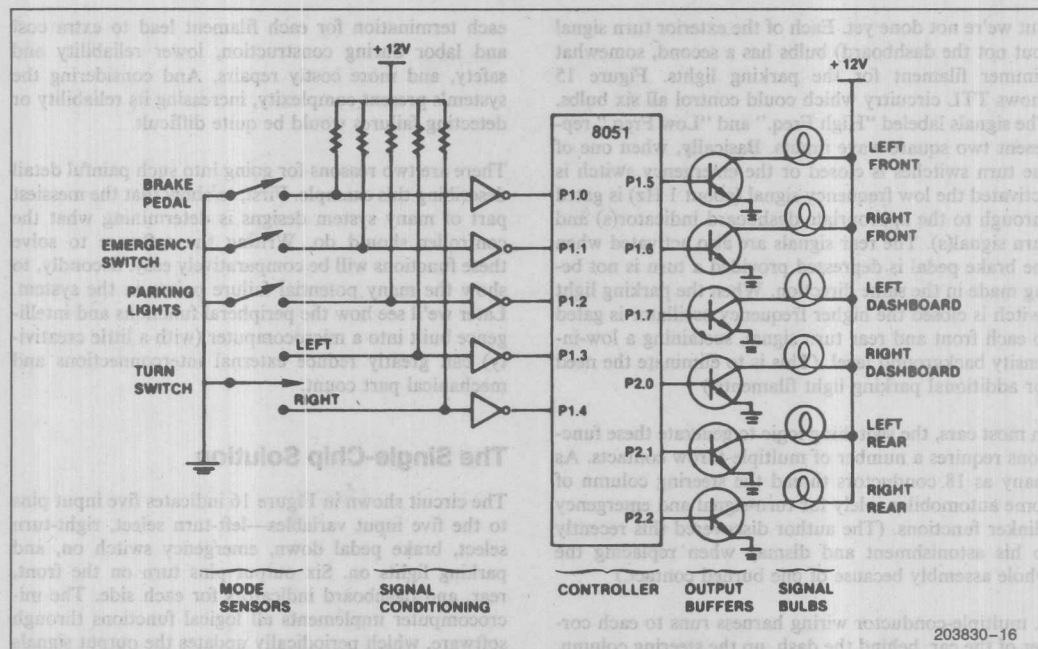


Figure 16. Microcomputer Turn-Signal Connections

Design Example #3 demonstrated that symbolic addressing with user-defined bit names makes code and documentation easier to write and maintain. Accordingly, we'll assign these I/O pins names for use throughout the program. (The format of this example will differ somewhat from the others. Segments of the overall program will be presented in sequence as each is described.)

```
R_DASH BIT P2.0 ;DASHBOARD RIGHT-
;TURN INDICATOR
I_REAR BIT P2.1 ;REAR LEFT-TURN
;INDICATOR
R_REAR BIT P2.2 ;REAR RIGHT-TURN
;INDICATOR
;
```

```
;
; INPUT PIN DECLARATIONS:
;(ALL INPUTS ARE POSITIVE-TRUE LOGIC)
;
BRAKE BIT P1.0 ;BRAKE PEDAL
;DEPRESSED
EMERG BIT P1.1 ;EMERGENCY BLINKER
;ACTIVATED
PARK BIT P1.2 ;PARKING LIGHTS ON
I_TURN BIT P1.3 ;TURN LEVER DOWN
R_TURN BIT P1.4 ;TURN LEVER UP
;
; OUTPUT PIN DECLARATIONS:
;
I_FRNT BIT P1.5 ;FRONT LEFT-TURN
;INDICATOR
R_FRNT BIT P1.6 ;FRONT RIGHT-TURN
;INDICATOR
I_DASH BIT P1.7 ;DASHBOARD LEFT-TURN
;INDICATOR
```

Another key advantage of symbolic addressing will appear further on in the design cycle. The locations of cable connectors, signal conditioning circuitry, voltage regulators, heat sinks, and the like all affect P.C. board layout. It's quite likely that the somewhat arbitrary pin assignment defined early in the software design cycle will prove to be less than optimum; rearranging the I/O pin assignment could well allow a more compact module, or eliminate costly jumpers on a single-sided board. (These considerations apply especially to automotive and other cost-sensitive applications needing single-chip controllers.) Since other architectures mask bytes or use "clever" algorithms to isolate bits by rotating them into the carry, re-routing an input signal (from bit 1 of port 1, for example, to bit 4 of port 3) could require extensive modifications throughout the software.

The Boolean Processor's direct bit addressing makes such changes absolutely trivial. The number of the port containing the pin is irrelevant, and masks and complex

program structures are not needed. Only the initial Boolean variable declarations need to be changed; ASM51 automatically adjusts all addresses and symbolic references to the reassigned variables. The user is assured that no additional debugging or software verification will be required.

```

; INTERRUPT RATE SUBDIVIDER
SUB_DIV DATA 20H
; HIGH-FREQUENCY OSCILLATOR BIT
HI_FREQ BIT SUB_DIV, 0
; LOW-FREQUENCY OSCILLATOR BIT
LO_FREQ BIT SUB_DIV, 7
;
; ORG 0000H
JMP INIT
;
; ORG 100H
; PUT TIMER 0 IN MODE 1
INIT: MOV TMOD, #00000001B
; INITIALIZE TIMER REGISTERS
MOV TLO, #0
MOV TH0, #-16
; SUBDIVIDE INTERRUPT RATE BY 244
MOV SUB_DIV, #244
; ENABLE TIMER INTERRUPTS
SETB ETO
; GLOBALLY ENABLE ALL INTERRUPTS
SETB EA
; START TIMER
SETB TRO
;
; (CONTINUE WITH BACKGROUND PROGRAM)
;
; PUT TIMER 0 IN MODE 1
; INITIALIZE TIMER REGISTERS

; SUBDIVIDE INTERRUPT RATE BY 244
; ENABLE TIMER INTERRUPTS
; GLOBALLY ENABLE ALL INTERRUPTS
; START TIMER

```

Timer 0 (one of the two on-chip timer counters) replaces the thermo-mechanical blinker relay in the dashboard controller. During system initialization it is configured as a timer in mode 1 by setting the least significant bit of the timer mode register (TMOD). In this configuration the low-order byte (TLO) is incremented every machine cycle, overflowing and incrementing the high-order byte (TH0) every 256 μ s. Timer interrupt 0 is enabled so that a hardware interrupt will occur each time TH0 overflows.

An eight-bit variable in the bit-addressable RAM array will be needed to further subdivide the interrupts via software. The lowest-order bit of this counter toggles very fast to modulate the parking lights: bit 7 will be

“tuned” to approximately 1 Hz for the turn- and emergency-indicator blinking rate. These three bits include a software variable and a hardware variable. Loading TH0 with -16 will cause an interrupt after 4.096 ms. The interrupt service routine reloads the high-order byte of timer 0 for the next interval, saves the CPU registers likely to be affected on the stack, and then decrements SUB_DIV. Loading SUB_DIV with 244 initially and each time it decrements to zero will produce a 0.999 second period for the highest-order bit.

```

ORG 000BH ; TIMER 0 SERVICE VECTOR
MOV TH0, #-16
PUSH PSW
PUSH ACC
PUSH B
DJNZ SUB_DIV, TOSERV
MOV SUB_DIV, #244

```

The code to sample inputs, perform calculations, and update outputs—the real “meat” of the signal controller algorithm—may be performed either as part of the interrupt service routine or as part of a background program loop. The only concern is that it must be executed at least several dozen times per second to prevent parking light flickering. We will assume the former case, and insert the code into the timer 0 service routine.

First, notice from the logic diagram (Figure 15) that the subterm (PARK • H_FREQ), asserted when the parking lights are to be on dimly, figures into four of the six output functions. Accordingly, we will first compute that term and save it in a temporary location named “DIM”. The PSW contains two general purpose flags: F0, which corresponds to the 8048 flag of the same name, and PSW.1. Since the PSW has been saved and will be restored to its previous state after servicing the interrupt, we can use either bit for temporary storage.

```

DIM BIT PSW.1 ; DECLARE TEMP
; ...
MOV C, PARK ; GATE PARKING
ANL HI_FREQ ; LIGHT SWITCH
; WITH HIGH
; FREQUENCY
; SIGNAL
MOV DIM, C ; AND SAVE IN
; TEMP. VARIABLE

```

This simple three-line section of code illustrates a remarkable point. The software indicates in very abstract terms exactly what function is being performed, inde-

pendent of the hardware configuration. The fact that these three bits include an input pin, a bit within a program variable, and a software flag in the PSW is totally invisible to the programmer.

Now generate and output the dashboard left turn signal.

```

;
MOV C,L_TURN      ;SET CARRY IF
                  ;TURN
ORL C,EMERG       ;OR EMERGENCY
ANL C,LO_FREQ     ;SELECTED
                  ;GATE IN 1 HZ
MOV I_DASH,C      ;SIGNAL
                  ;AND OUTPUT TO
                  ;DASHBOARD

```

To generate the left front turn signal we only need to add the parking light function in F0. But notice that the function in the carry will also be needed for the rear signal. We can save effort later by saving its current state in F0.

```

;
MOV F0,C          ;SAVE FUNCTION
                  ;SO FAR
ORL C,DIM         ;ADD IN PARKING
                  ;LIGHT FUNCTION
MOV L_FRNT,C      ;AND OUTPUT TO
                  ;TURN SIGNAL

```

Finally, the rear left turn signal should also be on when the brake pedal is depressed, provided a left turn is not in progress.

```

MOV C,BRAKE       ;GATE BRAKE
                  ;PEDAL SWITCH
ANL C,L_TURN      ;WITH TURN
                  ;LEVER
ORL C,F0          ;INCLUDE TEMP.
                  ;VARIABLE FROM DASH

```

```

ORL C,DIM         ;AND PARKING
                  ;LIGHT FUNCTION
MOV L_REAR,C      ;AND OUTPUT TO
                  ;TURN SIGNAL

```

Now we have to go through a similar sequence for the right-hand equivalents to all the left-turn lights. This also gives us a chance to see how the code segments above look when combined.

```

MOV C,R_TURN      ;SET CARRY H-
                  ;TURN
ORL C,EMERG       ;OR EMERGENCY
ANL C,LO_FREQ     ;SELECTED
                  ;IF SO, GATE IN 1
                  ;HZ SIGNAL
MOV R_DASH,C      ;AND OUTPUT TO
                  ;DASHBOARD
MOV F0,C          ;SAVE FUNCTION
                  ;SO FAR
ORL C,DIM         ;ADD IN PARKING
                  ;LIGHT FUNCTION
MOV R_FRNT,C      ;AND OUTPUT TO
                  ;TURN SIGNAL
MOV C,BRAKE       ;GATE BRAKE
                  ;PEDAL SWITCH
ANL C,R_TURN      ;WITH TURN
                  ;LEVER
ORL C,F0          ;INCLUDE TEMP.
                  ;VARIABLE FROM
                  ;DASH
ORL C,DIM         ;AND PARKING
                  ;LIGHT FUNCTION
MOV R_REAR,C      ;AND OUTPUT TO
                  ;TURN SIGNAL

```

(The perceptive reader may notice that simply rearranging the steps could eliminate one instruction from each sequence.)

Now that all six bulbs are in the proper states, we can return from the interrupt routine, and the program is finished. This code essentially needs to reverse the status saving steps at the beginning of the interrupt.

Table 7. Non-Trivial Duty Cycles

Sub_Div Bits									Duty Cycles						
7	6	5	4	3	2	1	0		12.5%	25.0%	37.5%	50.0%	62.5%	75.0%	87.5%
X	X	X	X	X	0	0	0		Off	Off	Off	Off	Off	Off	Off
X	X	X	X	X	0	0	1		Off	Off	Off	Off	Off	Off	On
X	X	X	X	X	0	1	0		Off	Off	Off	Off	Off	On	On
X	X	X	X	X	0	1	1		Off	Off	Off	Off	On	On	On
X	X	X	X	X	1	0	0		Off	Off	Off	On	On	On	On
X	X	X	X	X	1	0	1		Off	Off	On	On	On	On	On
X	X	X	X	X	1	1	0		Off	On	On	On	On	On	On
X	X	X	X	X	1	1	1		On	On	On	On	On	On	On

```

POP B ;RESTORE CPU
;REGISTERS.
POP ACC
POP PSW
RETI

```

Program Refinements. The luminescence of an incandescent light bulb filament is generally non-linear: the 50% duty cycle of `HL_FREQ` may not produce the desired intensity. If the application requires, duty cycles of 25%, 75%, etc. are easily achieved by ANDing and ORing in additional low-order bits of `SUB_DIV`. For example, 30 H/ signals of seven different duty cycles could be produced by considering bits 2-0 as shown in Table 7. The only software change required would be to the code which sets-up variable `DIM`;

```

MOV C,SUB_DIV.1;START WITH 50
;PERCENT
ANL C,SUB_DIV.0;MASK DOWN TO 25
;PERCENT
ORL C,SUB_DIV.2;AND BUILD BACK TO
;62 PERCENT
MOV DIM,C ;DUTY CYCLE FOR
;PARKING LIGHTS.

```

Interconnections increase cost and decrease reliability. The simple buffered pin-per-function circuit in Figure 16 is insufficient when many outputs require higher-than-TTL drive levels. A lower-cost solution uses the 8051 serial port in the shift-register mode to augment I/O. In mode 0, writing a byte to the serial port data buffer (SBUF) causes the data to be output sequentially through the "RXD" pin while a burst of eight clock pulses is generated on the "TXD" pin. A shift register connected to these pins (Figure 17) will load the data byte as it is shifted out. A number of special peripheral

driver circuits combining shift-register inputs with high drive level outputs have been introduced recently.

Cascading multiple shift registers end-to-end will expand the number of outputs even further. The data rate in the I/O expansion mode is one megabaud, or 8 μ s. per byte. This is the mode which the serial port defaults to following a reset, so no initialization is required.

The software for this technique uses the B register as a "map" corresponding to the different output functions. The program manipulates these bits instead of the output pins. After all functions have been calculated the B register is shifted by the serial port to the shift-register driver. (While some outputs may glitch as data is shifted through them, at 1 Megabaud most people wouldn't notice. Some shift registers provide an "enable" bit to hold the output states while new data is being shifted in.)

This is where the earlier decision to address bits symbolically throughout the program is going to pay off. This major I/O restructuring is nearly as simple to implement as rearranging the input pins. Again, only the bit declarations need to be changed.

```

I_FRNT BIT B.0 ;FRONT LEFT-TURN
;INDICATOR
R_FRNT BIT B.1 ;FRONT RIGHT-TURN
;INDICATOR
I_DASH BIT B.2 ;DASHBOARD LEFT-TURN
;INDICATOR
R_DASH BIT B.3 ;DASHBOARD RIGHT-TURN
;INDICATOR
I_REAR BIT B.4 ;REAR LEFT-TURN
;INDICATOR
R_REAR BIT B.5 ;REAR RIGHT-TURN
;INDICATOR

```

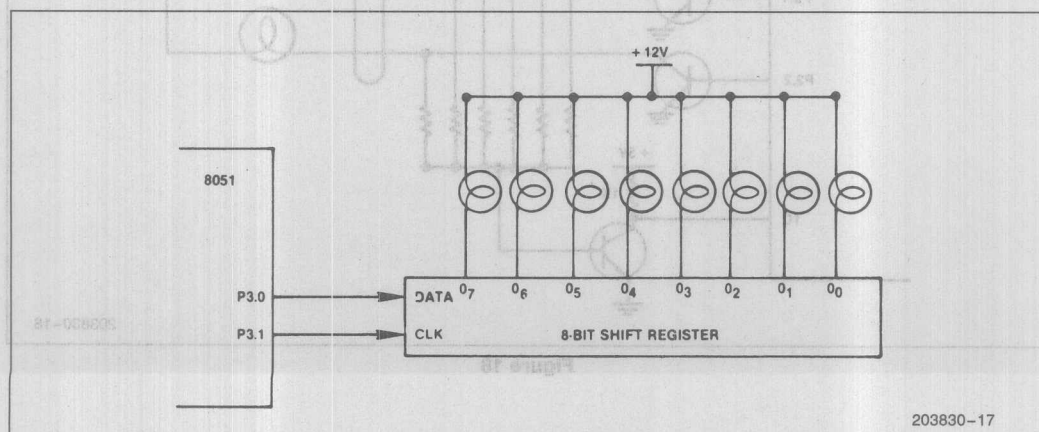


Figure 17. Output Expansion Using Serial Port

The original program to compute the functions need not change. After computing the output variables, the control map is transmitted to the buffered shift register through the serial port.

MOV SBUF,B ;LOAD BUFFER AND TRANSMIT

The Boolean Processor solution holds a number of advantages over older methods. Fewer switches are required. Each is simpler, requiring fewer poles and lower current contacts. The flasher relay is eliminated entirely. Only six filaments are driven, rather than 10. The wiring harness is therefore simpler and less expensive—one conductor for each of the six lamps and each of the five sensor switches. The fewer conductors use far fewer connectors. The whole system is more reliable.

And since the system is much simpler it would be feasible to implement redundancy and or fault detection on the four main turn indicators. Each could still be a

standard double filament bulb, but with the filaments driven in parallel to tolerate single-element failures.

Even with redundancy, the lights will eventually fail. To handle this inescapable fact current or voltage sensing circuits on each main drive wire can verify that each bulb and its high-current driver is functioning properly. Figure 18 shows one such circuit.

Assume all of the lights are turned on except one: i.e., all but one of the collectors are grounded. For the bulb which is turned off, if there is continuity from +12V through the bulb base and filament, the control wire, all connectors, and the P.C. board traces, and if the transistor is indeed not shorted to ground, then the collector will be pulled to +12V. This turns on the base of Q8 through the corresponding resistor, and grounds the input pin, verifying that the bulb circuit is operational. The continuity of each circuit can be checked by software in this way.

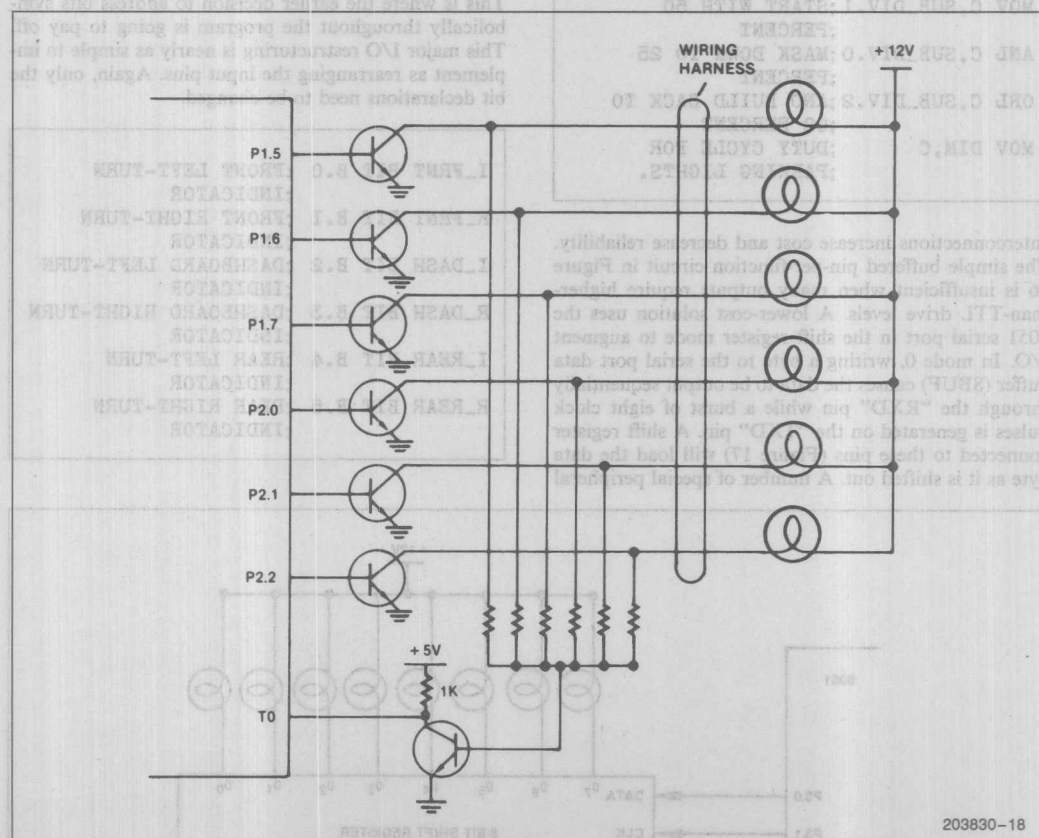


Figure 18

Now turn *all* the bulbs on, grounding all the collectors. Q7 should be turned off, and the Test pin should be high. However, a control wire shorted to +12V or an open-circuited drive transistor would leave one of the collectors at the higher voltage even now. This too would turn on Q7, indicating a different type of failure. Software could perform these checks once per second by executing the routine every time the software counter SUB_DIV is reloaded by the interrupt routine.

```

DJNZ SUB_DIV,TOSERV
MOV SUB_DIV,#244      ;RELOAD COUNTER
ORL P1,#11100000B     ;SET CONTROL
                       ;OUTPUTS HIGH

ORL P2,#00000111B
CLR I_FRNT            ;FLOAT DRIVE
                       ;COLLECTOR

JB TO,FAULT           ;TO SHOULD BE
                       ;PULLED LOW
SETB L_FRNT           ;PULL COLLECTOR
                       ;BACK DOWN

CLR L_DASH
JB TO,FAULT
SETB L_DASH
CLR L_REAR
JB TO,FAULT
SETB L_REAR
CLR R_FRNT
JB TO,FAULT
SETB R_FRNT
CLR R_DASH
JB TO,FAULT
SETB R_DASH
CLR R_REAR
JB TO,FAULT
SETB R_REAR

;
;WITH ALL COLLECTORS GROUNDED. TO
;SHOULD BE HIGH
;IF SO. CONTINUE WITH INTERRUPT
;ROUTINE.
JB TO,TOSERV
FAULT:                ;ELECTRICAL
                       ;FAILURE
                       ;PROCESSING
                       ;ROUTINE
                       ;(LEFT TO
                       ;READER'S
                       ;IMAGINATION)
TOSERV:                ;CONTINUE WITH
                       ;INTERRUPT
                       ;PROCESSING

```

The complete assembled program listing is printed in Appendix A. The resulting code consists of 67 program statements, not counting declarations and comments, which assemble into 150 bytes of object code. Each pass through the service routine requires (coincidentally) 67 μ s plus 32 μ s once per second for the electrical test. If executed every 4 ms as suggested this software would typically reduce the throughput of the background program by less than 2%.

Once a microcomputer has been designed into a system, new features suddenly become virtually free. Software could make the emergency blinkers flash alternately or at a rate faster than the turn signals. Turn signals could override the emergency blinkers. Adding more bulbs would allow multiple tail light sequencing and syncopation—true flash factor, so to speak.

Design Example #5—Complex Control Functions

Finally, we'll mix byte and bit operations to extend the use of 8051 into extremely complex applications.

Programmers can arbitrarily assign I/O pins to input and output functions only if the total does not exceed 32, which is insufficient for applications with a very large number of input variables. One way to expand the number of inputs is with a technique similar to multiplexed-keyboard scanning.

Figure 19 shows a block diagram for a moderately complex programmable industrial controller with the following characteristics:

- 64 input variable sensors:
- 12 output signals:
- Combinational and sequential logic computations:
- Remote operation with communications to a host processor via a high-speed full-duplex serial link:
- Two prioritized external interrupts:
- Internal real-time and time-of-day clocks.

While many microprocessors could be programmed to provide these capabilities with assorted peripheral support chips, an 8051 microcomputer needs no other integrated circuits!

The 64 input sensors are logically arranged as an 8x8 matrix. The pins of Port 1 sequentially enable each column of the sensor matrix: as each is enabled Port 0 reads in the state of each sensor in that column. An eight-byte block in bit-addressable RAM remembers the data as it is read in so that after each complete scan cycle there is an internal map of the current state of all sensors. Logic functions can then directly address the elements of the bit map.

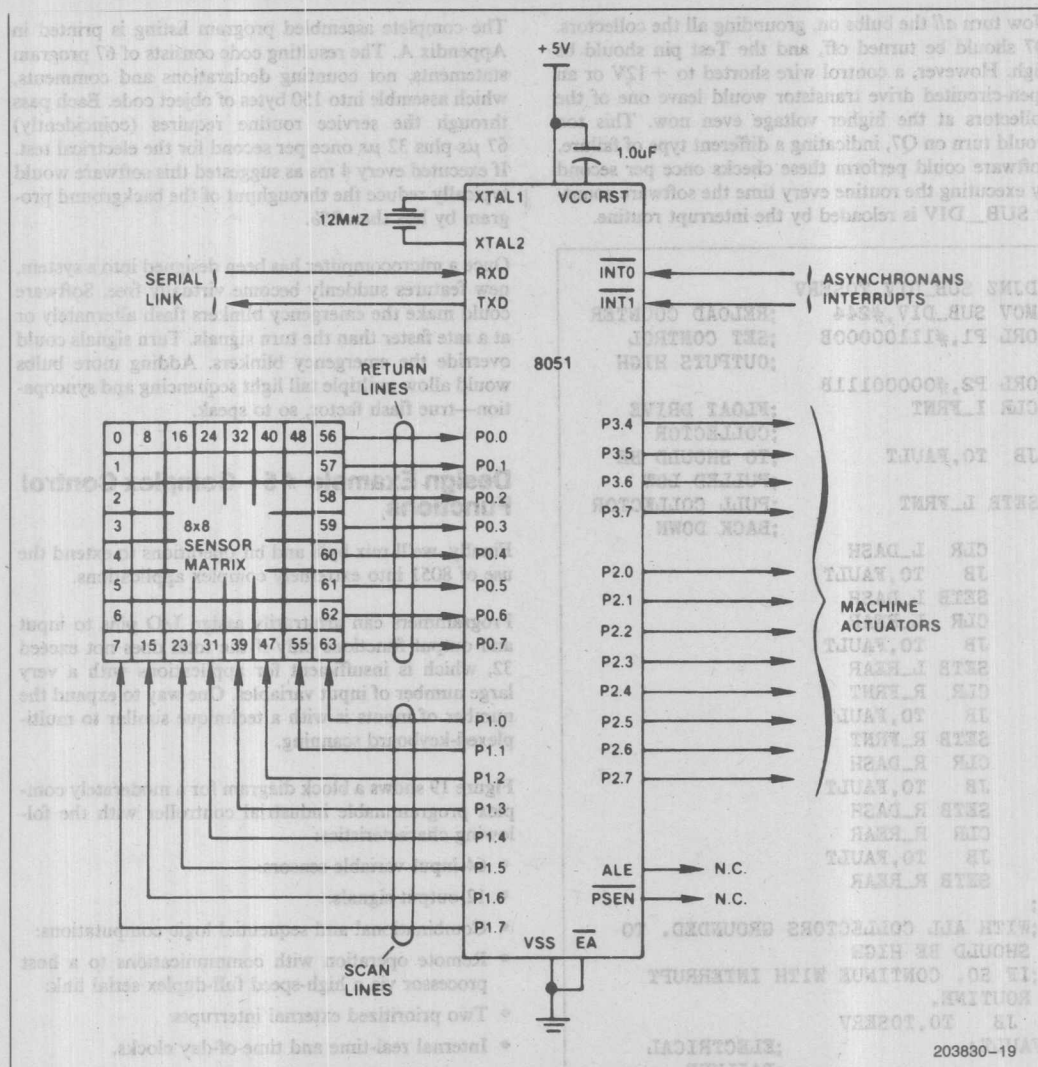


Figure 19. Block Diagram of 64-Input Machine Controller

The computer's serial port is configured as a nine-bit UART, transferring data at 17,000 bytes-per-second. The ninth bit may distinguish between address and data bytes.

The 8051 serial port can be configured to detect bytes with the address bit set, automatically ignoring all others. Pins INT0 and INT1 are interrupts configured respectively as high-priority, falling-edge triggered and low-priority, low-level triggered. The remaining 12 I/O pins output TTL-level control signals to 12 actuators.

There are several ways to implement the sensor matrix circuitry, all logically similar. Figure 20a shows one possibility. Each of the 64 sensors consists of a pair of simple switch contacts in series with a diode to permit multiple contact closures throughout the matrix.

The scan lines from Port 1 provide eight un-encoded active-high scan signals for enabling columns of the matrix. The return lines on rows where a contact is closed are pulled high and read as logic ones. Open return lines are pulled to ground by one of the 40 kΩ resistors and are read as zeroes. (The resistor values must be chosen to ensure all return lines are pulled above the 2.0V logic threshold, even in the worst-case,

where all contacts in an enabled column are closed.) Since P0 is provided open-collector outputs and high-impedance MOS inputs its input loading may be considered negligible.

The circuits in Figures 20b-20d are variations on this theme. When input signals must be electrically isolated from the computer circuitry as in noisy industrial environments, phototransistors can replace the switch diode pairs and provide optical isolation as in Figure 20b. Additional opto-isolators could also be used on the control output and special signal lines.

The other circuits assume that input signals are already at TTL levels. Figure 20c uses octal three-state buffers enabled by active-low scan signals to gate eight signals onto Port 0. Port 0 is available for memory expansion or peripheral chip interfacing between sensor matrix scans. Eight-to-one multiplexers in Figure 20d select one of eight inputs for each return line as determined by encoded address bits output on three pins of Port 1. (Five more output pins are thus freed for more control functions.) Each output can drive at least one standard TTL or up to 10 low-power TTL loads without additional buffering.

Going back to the original matrix circuit, Figure 21 shows the method used to scan the sensor matrix. Two complete bit maps are maintained in the bit-addressable region of the RAM: one for the current state and one for the previous state read for each sensor. If the need arises, the program could then sense input transitions and or debounce contact closures by comparing each bit with its earlier value.

The code in Example 3 implements the scanning algorithm for the circuits in Figure 20a. Each column is enabled by setting a single bit in a field of zeroes. The bit maps are positive logic: ones represent contacts that are closed or isolators turned on.

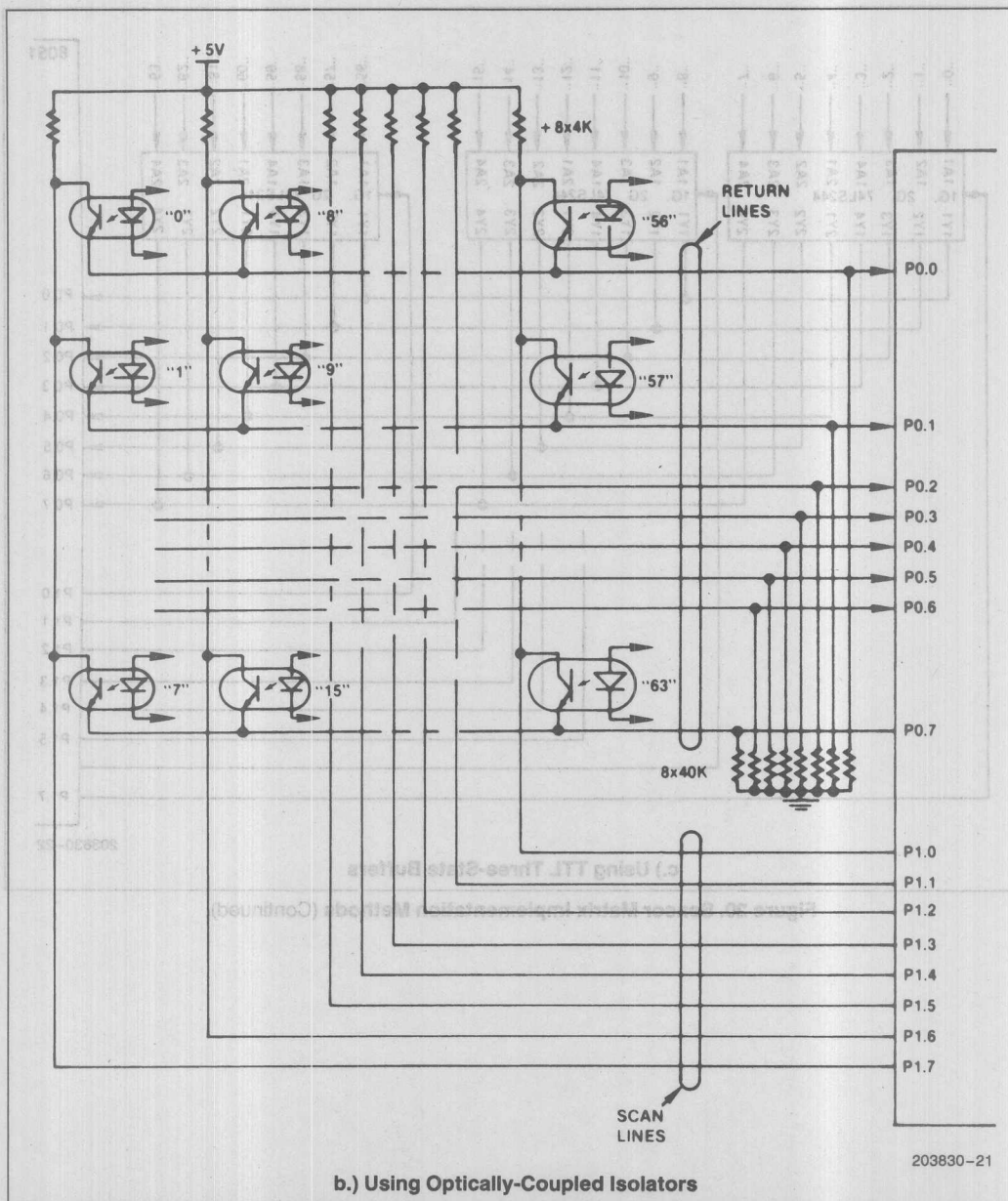
Example 3.

```

INPUT_SCAN:      ;SUBROUTINE TO READ
                  ;CURRENT STATE
                  ;OF 64 SENSORS AND
                  ;SAVE IN RAM 20H-27H
MOV R0,#20H      ;INITIALIZE
                  ;POINTERS
MOV R1,#28H      ;FOR BIT MAP
MOV A,#80H       ;BASES
                  ;SET FIRST BIT
                  ;IN ACC
SCAN: MOV P1,A    ;OUTPUT TO SCAN
                  ;LINES
RR A             ;SHIFT TO ENABLE
                  ;NEXT COLUMN
                  ;NEXT
MOV R2,A         ;REMEMBER CUR-
                  ;RENT SCAN
                  ;POSITION
MOV A,P0         ;READ RETURN
                  ;LINES
XCH A,@R0        ;SWITCH WITH
                  ;PREVIOUS MAP
                  ;BITS
MOV @R1,A        ;SAVE PREVIOUS
                  ;STATE AS WELL
INC R0           ;BUMP POINTERS
INC R1
MOV A,R2         ;RELOAD SCAN
                  ;LINE MASK
JNB ACC,7;SCAN; ;LOOP UNTIL ALL
                  ;EIGHT COLUMNS
                  ;READ
RET

```

Figure 20. Sensor Matrix Implementation Methods



b.) Using Optically-Coupled Isolators

Figure 20. Sensor Matrix Implementation Methods (Continued)

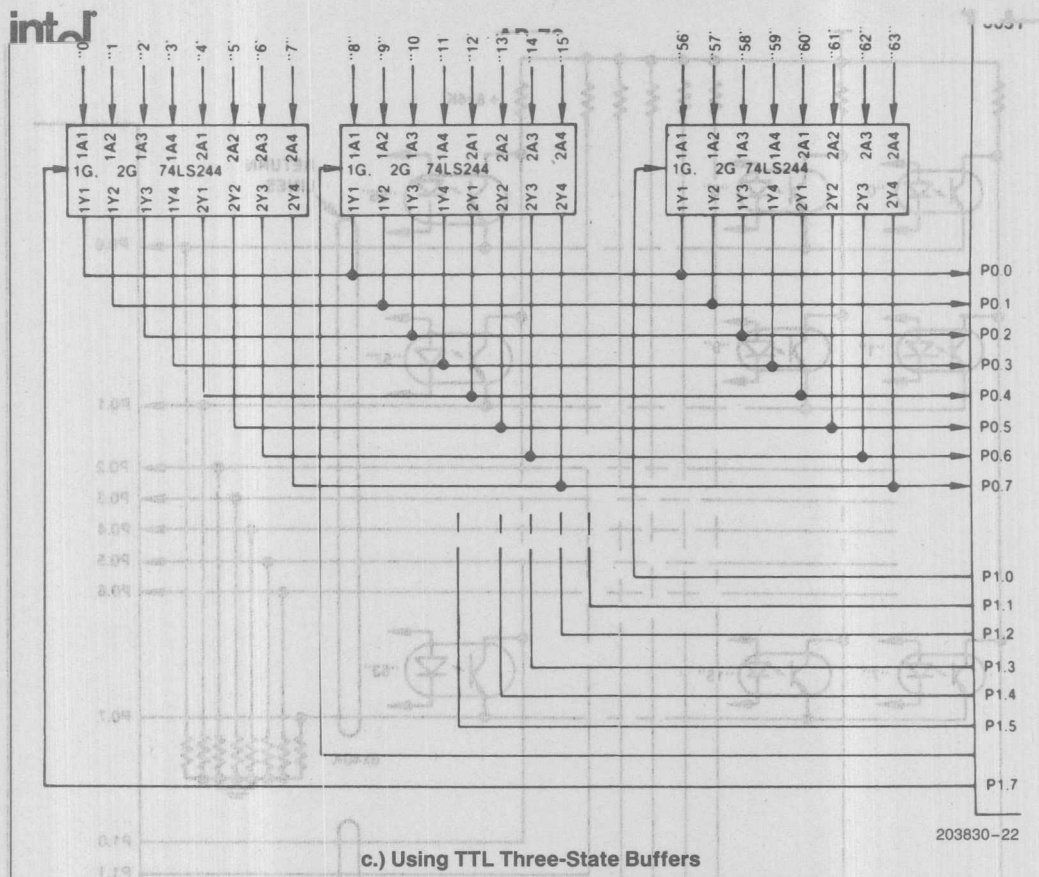
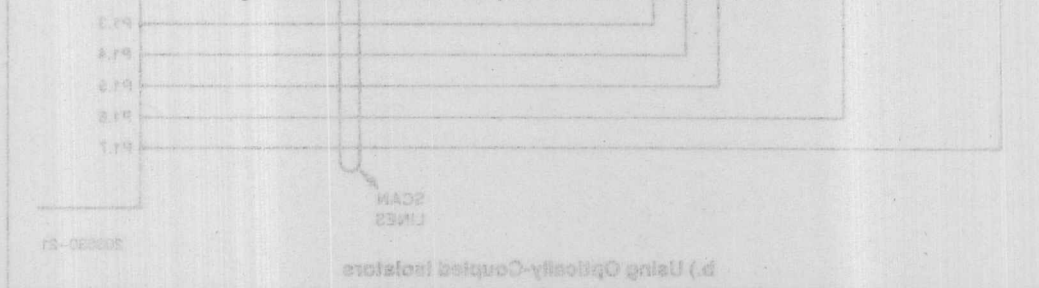


Figure 20. Sensor Matrix Implementation Methods (Continued)



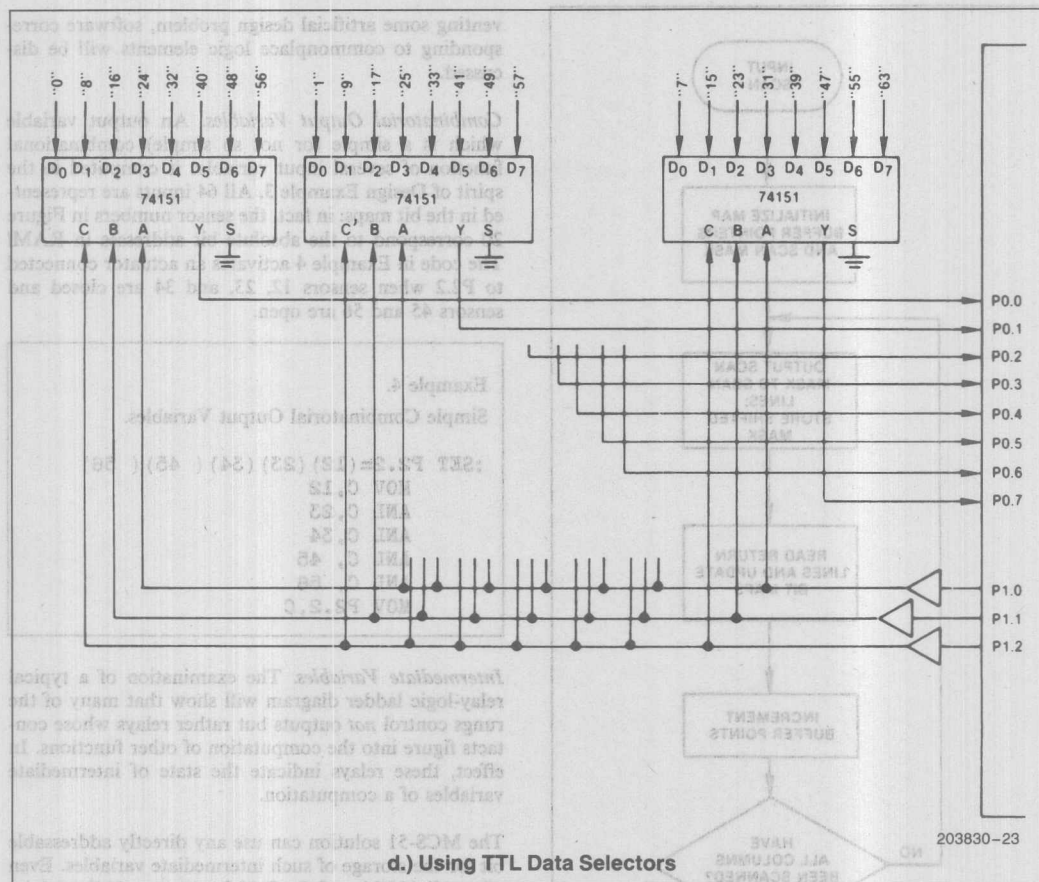


Figure 20. Sensor Matrix Implementation Methods (Continued)

Figure 21. Flowchart for Reading in Sensor Matrix

What happens after the sensors have been scanned depends on the individual application. Rather than in-

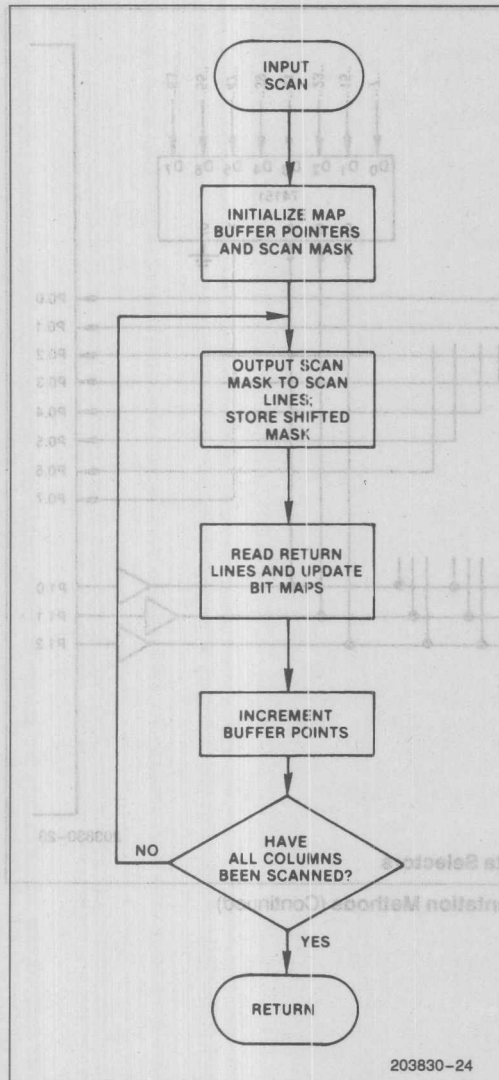


Figure 21. Flowchart for Reading in Sensor Matrix

What happens after the sensors have been scanned depends on the individual application. Rather than in-

venting some artificial design problem, software corresponding to commonplace logic elements will be discussed.

Combinatorial Output Variables. An output variable which is a simple (or not so simple) combinational function of several input variables is computed in the spirit of Design Example 3. All 64 inputs are represented in the bit maps: in fact, the sensor numbers in Figure 20 correspond to the absolute bit addresses in RAM! The code in Example 4 activates an actuator connected to P2.2 when sensors 12, 23, and 34 are closed and sensors 45 and 56 are open.

Example 4.

Simple Combinatorial Output Variables.

```

;SET P2.2=(12) (23) (34) ( 45) ( 56)
MOV C,12
ANL C,23
ANL C,34
ANL C, 45
ANL C, 56
MOV P2.2,C
  
```

Intermediate Variables. The examination of a typical relay-logic ladder diagram will show that many of the rungs control *not* outputs but rather relays whose contacts figure into the computation of other functions. In effect, these relays indicate the state of intermediate variables of a computation.

The MCS-51 solution can use any directly addressable bit for the storage of such intermediate variables. Even when all 128 bits of the RAM array are dedicated (to input bit maps in this example), the accumulator, PSW, and B register provide 18 additional flags for intermediate variables.

For example, suppose switches 0 through 3 control a safety interlock system. Closing any of them should deactivate certain outputs. Figure 22 is a ladder diagram for this situation. The interlock function could be recomputed for every output affected, or it may be computed once and save (as implied by the diagram). As the program proceeds this bit can qualify each output.

Example 5. Incorporating Override signal into actuator outputs.

```

; CALL INPUT_SCAN
MOV C,0
ORL C,1
ORL C,2
ORL C,3
MOV FO,C
; .....
; COMPUTE FUNCTION 0
;
ANL C, FO
MOV PLO,C
; .....
; COMPUTE FUNCTION 1
;
ANL C, FO
MOV P1,1,C
; .....
; COMPUTE FUNCTION 2
;
ANL C, FO
MOV P1,2,C
; .....

```

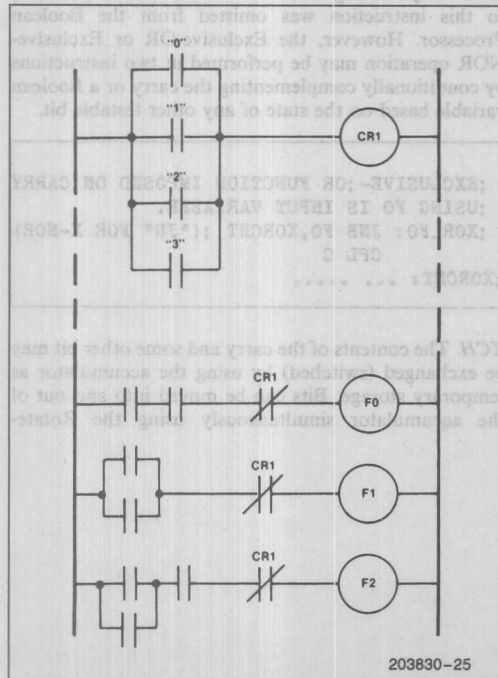


Figure 22. Ladder Diagram for Output Override Circuitry

Latching Relays. A latching relay can be forced into either the ON or OFF state by two corresponding input signals, where it will remain until forced onto the opposite state—analogueous to a TTL Set/Reset flip-flop. The relay is used as an intermediate variable for other calculations. In the previous example, the emergency condition could be remembered and remain active until an "emergency cleared" button is pressed.

Any flag or addressable bit may represent a latching relay with a few lines of code (see Example 6).

Example 6. Simulating a latching relay.

```

;I_SET SET FLAG 0 IF C=1
I_SET: ORL C,FO
      MOV FO,C
;
;I_RSET RESET FLAG 0 IF C=1
I_RSET: CPL C
      ANL C,FO
      MOV FO,C
;

```

Time Delay Relays. A time delay relay does not respond to an input signal until it has been present (or absent) for some predefined time. For example, a ballast or load resistor may be switched in series with a D.C. motor when it is first turned on, and shunted from the circuit after one second. This sort of time delay may be simulated by an interrupt routine driven by one of the two 8051 timer counters. The procedure followed by the routine depends heavily on the details of the exact function needed: time-outs or time delays with resettable or non-resettable inputs are possible. If the interrupt routine is executed every 10 milliseconds the code in Example 7 will clear an intermediate variable set by the background program after it has been active for two seconds.

Example 7. Code to clear USRFLG after a fixed time delay.

```

JNB USR_FLG,NXTTST
DJNZ DLAY_COUNT,NXTTST
CLR USR_FLG
MOV DLAY_COUNT,#200
NXTTST; ;...

```


Here the Boolean processor pays off. *Every instruction mentioned in this Note* completes in one or two microseconds—the *minimum* instruction execution time for many other microcontrollers! A ladder diagram containing a hundred rungs, with an average of four contacts per rung can be replaced by approximately five hundred lines of software. A complete pass through the entire matrix scanning routine and all computations would require about a millisecond: less than the time it takes for most relays to change state.

an function with a subroutine would be less efficient by at least an order of magnitude. Extra software is needed for the simulation routines, and each step takes longer to execute for three reasons: several byte-wide logical instructions are executed per user program step (rather than one Boolean operation); most of those instructions take longer to execute with microprocessors performing multiple off-chip accesses; and calling and returning from the various subroutines requires overhead for stack operations.

In fact, the speed of the Boolean Processor solution is likely to be much faster than the system requires. The CPU might use the time left over to compute feedback parameters, collect and analyze execution statistics, perform system diagnostics, and so forth.

Additional Functions and Uses

With the building-block basics mentioned above many more operations may be synthesized by short instruction sequences.

Exclusive-OR. There are no common mechanical devices or relays analogous to the Exclusive-OR operation, so this instruction was omitted from the Boolean Processor. However, the Exclusive-OR or Exclusive-NOR operation may be performed in two instructions by conditionally complementing the carry or a Boolean variable based on the state of any other testable bit.

```

;EXCLUSIVE-;OR FUNCTION IMPOSED ON CARRY
;USING FO IS INPUT VARIABLE.
;XOR_FO: JNB FO,XORCNT ;("JB" FOR X-NOR)
        CPL C
;XORCNT: ... ..

```

XCH. The contents of the carry and some other bit may be exchanged (switched) by using the accumulator as temporary storage. Bits can be moved into and out of the accumulator simultaneously using the Rotate-

through-carry instructions, though this would alter the accumulator data.

```

;EXCHANGE CARRY WITH USRFLG
XCHBIT: RLC    A
        MOV    C,USR_FLG
        RRC    A
        MOV    USR_FLG,C
        RLC    A

```

Extended Bit Addressing. The 8051 can directly address 144 general-purpose bits for all instructions in Figure 3b. Similar operations may be extended to any bit anywhere on the chip with some loss of efficiency.

The logical operations AND, OR, and Exclusive-OR are performed on byte variables using six different addressing modes, one of which lets the source be an immediate mask, and the destination any directly addressable byte. Any bit may thus be set, cleared, or complemented with a three-byte, two-cycle instruction if the mask has all bits but one set or cleared.

Byte variables, registers, and indirectly addressed RAM may be moved to a bit addressable register (usually the accumulator) in one instruction. Once transferred, the bits may be tested with a conditional jump, allowing any bit to be polled in 3 microseconds—still much faster than most architectures—or used for logical calculations. (This technique can also simulate additional bit addressing modes with byte operations.)

Parity of bytes or bits. The parity of the current accumulator contents is always available in the PSW, from whence it may be moved to the carry and further processed. Error-correcting Hamming codes and similar applications require computing parity on groups of isolated bits. This can be done by conditionally complementing the carry flag based on those bits or by gathering the bits into the accumulator (as shown in the DES example) and then testing the parallel parity flag.

Multiple byte shift and CRC codes

Though the 8051 serial port can accommodate eight- or nine-bit data transmissions, some protocols involve much longer bit streams. The algorithms presented in

Design Example 2 can be extended quite readily to 16 or more bits by using multi-byte input and output buffers.

Many mass data storage peripherals and serial communications protocols include Cyclic Redundancy (CRC) codes to verify data integrity. The function is generally computed serially by hardware using shift registers and Exclusive-OR gates, but it can be done with software. As each bit is received into the carry, appropriate bits in the multi-byte data buffer are conditionally complemented based on the incoming data bit. When finished, the CRC register contents may be checked for zero by ORing the two bytes in the accumulator.

4.0 SUMMARY

A truly unique facet of the Intel MCS-51 microcomputer family design is the collection of features optimized for the one-bit operations so often desired in real-world, real-time control applications. Included are 17 special instructions, a Boolean accumulator, implicit and direct addressing modes, program and mass data storage, and many I/O options. These are the world's first single-chip microcomputers able to efficiently manipulate, operate on, and transfer either bytes or individual bits as data.

This Application Note has detailed the information needed by a microcomputer system designer to make full use of these capabilities. Five design examples were used to contrast the solutions allowed by the 8051 and those required by previous architectures. Depending on the individual application, the 8051 solution will be easier to design, more reliable to implement, debug, and verify, use less program memory, and run up to an order of magnitude faster than the same function implemented on previous digital computer architectures.

Combining byte- and bit-handling capabilities in a single microcomputer has a strong synergistic effect: the power of the result exceeds the power of byte- and bit-processors laboring individually. Virtually all user applications will benefit in some way from this duality. Data intensive applications will use bit addressing for test pin monitoring or program control flags; control applications will use byte manipulation for parallel I/O expansion or arithmetic calculations.

It is hoped that these design examples give the reader an appreciation of these unique features and suggest ways to exploit them in his or her own application.

APPENDIX A Automobile Turn-Indicator Controller Program Listing

ISIS-II MCS-51 MACRO ASSEMBLER V1.0
OBJECT MODULE PLACED IN FO AP70 HEX
ASSEMBLER INVOKED BY: f1.asm51 ap70 src date(328)

LOC	OBJ	LINE	SOURCE
		1	*XREF TITLE(AP-70 APPENDIX)
		2	*****
		3	
		4	THE FOLLOWING PROGRAM USES THE BOOLEAN INSTRUCTION SET
		5	OF THE INTEL 8051 MICROCOMPUTER TO PERFORM A NUMBER OF
		6	AUTOMOTIVE DASHBOARD CONTROL FUNCTIONS RELATING TO
		7	TURN SIGNAL CONTROL, EMERGENCY BLINKERS, BRAKE LIGHT
		8	CONTROL, AND PARKING LIGHT OPERATION.
		9	THE ALGORITHMS AND HARDWARE ARE DESCRIBED IN DESIGN
		10	EXAMPLE #4 OF INTEL APPLICATION NOTE AP-70.
		11	"USING THE INTEL MCS-51(TM)
		12	BOOLEAN PROCESSING CAPABILITIES"
		13	*****
		14	
		15	
		16	INPUT PIN DECLARATIONS:
		17	(ALL INPUTS ARE POSITIVE-TRUE LOGIC
		18	INPUTS ARE HIGH WHEN RESPECTIVE SWITCH CONTACT IS CLOSED)
		19	
0090		20	BRAKE BIT P1.0 ; BRAKE PEDAL DEPRESSED
0091		21	EMERG BIT P1.1 ; EMERGENCY BLINKER ACTIVATED
0092		22	PARK BIT P1.2 ; PARKING LIGHTS ON
0093		23	L_TURN BIT P1.3 ; TURN LEVER DOWN
0094		24	R_TURN BIT P1.4 ; TURN LEVER UP
		25	
		26	OUTPUT PIN DECLARATIONS:
		27	(ALL OUTPUTS ARE POSITIVE TRUE LOGIC
		28	BULB IS TURNED ON WHEN OUTPUT PIN IS HIGH.)
		29	
0095		30	L_FRNT BIT P1.5 ; FRONT LEFT-TURN INDICATOR
0096		31	R_FRNT BIT P1.6 ; FRONT RIGHT-TURN INDICATOR
0097		32	L_DASH BIT P1.7 ; DASHBOARD LEFT-TURN INDICATOR
00A0		33	R_DASH BIT P2.0 ; DASHBOARD RIGHT-TURN INDICATOR
00A1		34	L_REAR BIT P2.1 ; REAR LEFT-TURN INDICATOR
00A2		35	R_REAR BIT P2.2 ; REAR RIGHT-TURN INDICATOR
		36	
00A3		37	S_FAIL BIT P2.3 ; ELECTRICAL SYSTEM FAULT INDICATOR
		38	
		39	INTERNAL VARIABLE DEFINITIONS:
		40	
0020		41	SUB_DIV DATA 20H ; INTERRUPT RATE SUBDIVIDER
0000		42	HI_FREQ BIT SUB_DIV.0 ; HIGH-FREQUENCY OSCILLATOR BIT
0007		43	LO_FREQ BIT SUB_DIV.7 ; LOW-FREQUENCY OSCILLATOR BIT
		44	
00D1		45	DIM BIT PSW.1 ; PARKING LIGHTS ON FLAG
		46	
		47	*****
		48	\$EJECT

LOC	OBJ	LINE	SOURCE	END	COMMENT
		49	ORG	0000H	RESET VECTOR
0000	020040	50	LJMP	INIT	INITIALIZE
		51			
000B		52	ORG	000BH	TIMER 0 SERVICE VECTOR
000B	758CF0	53	MOV	TH0, #-16	HIGH TIMER BYTE ADJUSTED TO CONTROL INT. RATE
000E	C0D0	54	PUSH	PSW	EXECUTE CODE TO SAVE ANY REGISTERS USED BELOW
0010	0154	55	AJMP	UPDATE	UPDATE (CONTINUE WITH REST OF ROUTINE)
		56			
0040		57	ORG	0040H	INITIALIZE
0040	758A00	58	MOV	TLO, #0	ZERO LOADED INTO LOW-ORDER BYTE AND
0043	758CF0	59	MOV	TH0, #-16	-16 IN HIGH-ORDER BYTE GIVES 4 MSEC PERIOD
0046	758961	60	MOV	TMOD, #01100001B	8-BIT AUTO RELOAD COUNTER MODE FOR TIMER 1,
		61			16-BIT TIMER MODE FOR TIMER 0 SELECTED
0049	7520F4	62	MOV	SUB_DIV, #244	SUBDIVIDE INTERRUPT RATE BY 244 FOR 1 HZ
004C	D2A9	63	SETB	ETC	USE TIMER 0 OVERFLOWS TO INTERRUPT PROGRAM
004E	D2AF	64	SETB	EA	CONFIGURE IE TO GLOBALLY ENABLE INTERRUPTS
0050	D2BC	65	SETB	TRO	KEEP INSTRUCTION CYCLE COUNT UNTIL OVERFLOW
0052	80FE	66	SJMP	\$	START BACKGROUND PROGRAM EXECUTION
		67			
		68			
0054	D5203B	69	UPDATE: DJNZ	SUB_DIV, TOSERV	EXECUTE SYSTEM TEST ONLY ONCE PER SECOND
0057	7520F4	70	MOV	SUB_DIV, #244	GET VALUE FOR NEXT ONE SECOND DELAY AND
		71			GO THROUGH ELECTRICAL SYSTEM TEST CODE:
005A	4390E0	72	ORL	P1, #11100000B	SET CONTROL OUTPUTS HIGH
005D	43A007	73	ORL	P2, #00000111B	ONLY DRIVE LEDS ON
0060	C295	74	CLR	L_FRNT	FLOAT DRIVE COLLECTOR
0062	20B42B	75	JB	TO, FAULT	TO SHOULD BE PULLED LOW
0065	D295	76	SETB	L_FRNT	PULL COLLECTOR BACK DOWN
0067	C297	77	CLR	L_DASH	REPEAT SEQUENCE FOR L_DASH,
0069	20B421	78	JB	TO, FAULT	VD IN WORKING FIGHT POSITION
006C	D297	79	SETB	L_DASH	SAVE POSITION TO LVR
006E	C2A1	80	CLR	L_REAR	L_REAR,
0070	20B41A	81	JB	TO, FAULT	TO SHOULD BE PULLED LOW
0073	D2A1	82	SETB	L_REAR	PULL COLLECTOR BACK DOWN
0075	C296	83	CLR	R_FRNT	REPEAT SEQUENCE FOR R_FRNT,
0077	20B413	84	JB	TO, FAULT	TO SHOULD BE PULLED LOW
007A	D296	85	SETB	R_FRNT	PULL COLLECTOR BACK DOWN
007C	C2A0	86	CLR	R_DASH	REPEAT SEQUENCE FOR R_DASH,
007E	20B40C	87	JB	TO, FAULT	TO SHOULD BE PULLED LOW
0081	D2A0	88	SETB	R_DASH	PULL COLLECTOR BACK DOWN
0083	C2A2	89	CLR	R_REAR	REPEAT SEQUENCE FOR R_REAR,
0085	20B405	90	JB	TO, FAULT	TO SHOULD BE PULLED LOW
008B	D2A2	91	SETB	R_REAR	PULL COLLECTOR BACK DOWN
		92			
00A1	8500	93			WITH ALL COLLECTORS GROUNDED, TO SHOULD BE HIGH
00B1	V501	94			IF SO, CONTINUE WITH INTERRUPT ROUTINE
		95			
00BA	20B402	96	JB	TO, TOSERV	TO SHOULD BE PULLED LOW
00BD	B2A3	97	FAULT: CPL	S_FAIL	ELECTRICAL FAILURE PROCESSING ROUTINE
		98			(TOGGLE INDICATOR ONCE PER SECOND)
		99	+1	\$EJECT	

LOC	OBJ	LINE	SOURCE
		100	CONTINUE WITH INTERRUPT PROCESSING (INDICATION ONCE PER SECOND)
008D 8592		101	
008E 508+05		102	1) COMPUTE LOW BULB INTENSITY WHEN PARKING LIGHTS ARE ON.
		103	
008F A201		104	TOSERV: MOV C, SUB_DIV 1 ; START WITH 50 PERCENT.
0091 8200		105	ANL C, SUB_DIV 0 ; MASK DOWN TO 25 PERCENT.
0093 7202		106	ORL C, SUB_DIV 2 ; BUILD BACK TO 62.5 PERCENT.
0095 8292		107	ANL C, PARK ; GATE WITH PARKING LIGHT SWITCH.
0097 92D1		108	MOV DIM, C ; AND SAVE IN TEMP. VARIABLE.
		109	
0091 8592		110	2) COMPUTE AND OUTPUT LEFT-HAND DASHBOARD INDICATOR
0092 508+0C		111	
0099 A293		112	MOV C, L_TURN ; SET CARRY IF TURN
009B 7291		113	ORL C, EMERG ; OR EMERGENCY SELECTED.
009D 8207		114	ANL C, LO_FREQ ; IF SO, GATE IN 1 HZ SIGNAL
009F 9297		115	MOV L_DASH, C ; AND OUTPUT TO DASHBOARD.
		116	
00A0 508+1V		117	3) COMPUTE AND OUTPUT LEFT-HAND FRONT TURN SIGNAL.
009E 8597		118	
00A1 92D5		119	MOV F0, C ; SAVE FUNCTION SO FAR.
00A3 72D1		120	ORL C, DIM ; ADD IN PARKING LIGHT FUNCTION
00A5 9295		121	MOV L_FRNT, C ; AND OUTPUT TO TURN SIGNAL.
		122	
0095 508+3B		123	4) COMPUTE AND OUTPUT LEFT-HAND REAR TURN SIGNAL.
009D 8592		124	
00A7 A290		125	MOV C, BRAKE ; GATE BRAKE PEDAL SWITCH
00A9 B093		126	ANL C, L_TURN ; WITH TURN LEVER
00AB 72D5		127	ORL C, F0 ; INCLUDE TEMP. VARIABLE FROM DASH
00AD 72D1		128	ORL C, DIM ; AND PARKING LIGHT FUNCTION
00AF 92A1		129	MOV L_REAR, C ; AND OUTPUT TO TURN SIGNAL
		130	
		131	5) REPEAT ALL OF ABOVE FOR RIGHT-HAND COUNTERPARTS.
		132	
0081 A294		133	MOV C, R_TURN ; SET CARRY IF TURN
0083 7291		134	ORL C, EMERG ; OR EMERGENCY SELECTED
0085 8207		135	ANL C, LO_FREQ ; IF SO, GATE IN 1 HZ SIGNAL
0087 92A0		136	MOV R_DASH, C ; AND OUTPUT TO DASHBOARD
0089 92D5		137	MOV F0, C ; SAVE FUNCTION SO FAR
008B 72D1		138	ORL C, DIM ; ADD IN PARKING LIGHT FUNCTION
008D 9296		139	MOV R_FRNT, C ; AND OUTPUT TO TURN SIGNAL
008F A290		140	MOV C, BRAKE ; GATE BRAKE PEDAL SWITCH
00C1 B094		141	ANL C, R_TURN ; WITH TURN LEVER
00C3 72D5		142	ORL C, F0 ; INCLUDE TEMP. VARIABLE FROM DASH
00C5 72D1		143	ORL C, DIM ; AND PARKING LIGHT FUNCTION
00C7 92A2		144	MOV R_REAR, C ; AND OUTPUT TO TURN SIGNAL
		145	
0082 128C6		146	RESTORE STATUS REGISTER AND RETURN
008E		147	
00C9 D0D0		148	POP PSW ; RESTORE PSW
00CB 32		149	RETI ; AND RETURN FROM INTERRUPT ROUTINE
		150	
00C 087		151	END

WORLDWIDE MOTOR
ENGINEERING SOLUTIONS

XREF SYMBOL TABLE LISTING

NAME	TYPE	VALUE AND REFERENCES
BRAKE	N BSEG	0090H 20# 125 140
DIM	N BSEG	00D1H 45# 108 120 128 138 143
EA	N BSEG	00AFH 64
EMERG	N BSEG	0091H 21# 113 134
ETO	N BSEG	00A9H 63
F0	N BSEG	00D5H 119 127 137 142
FAULT	L CSEG	00BDH 75 78 81 84 87 90 97#
HI_FREQ	N BSEG	0000H 42#
INIT	L CSEG	0040H 50 58#
L_DASH	N BSEG	0097H 32# 77 79 115
L_FRNT	N BSEG	0095H 30# 74 76 121
L_REAR	N BSEG	00A1H 34# 80 82 129
L_TURN	N BSEG	0093H 23# 112 126
LO_FREQ	N BSEG	0007H 43# 114 135
P1	N DSEG	0090H 20 21 22 23 24 30 31 32 72
P2	N DSEG	00A0H 33 34 35 37 73
PARK	N BSEG	0092H 22# 107
PSW	N DSEG	00D0H 45 54 148
R_DASH	N BSEG	00A0H 33# 86 88 136
R_FRNT	N BSEG	0096H 31# 83 85 139
R_REAR	N BSEG	00A2H 35# 89 91 144
R_TURN	N BSEG	0094H 24# 133 141
S_FAIL	N BSEG	00A3H 37# 97
SUB_DIV	N DSEG	0020H 41# 42 43 62 69 70 104 105 106
T0	N BSEG	00B4H 75 78 81 84 87 90 96
TOSERV	L CSEG	00BFH 69 96 104#
TH0	N DSEG	00BCH 53 59
TLO	N DSEG	00BAH 58
TMOD	N DSEG	00B9H 60
TRO	N BSEG	00BCH 65
UPDATE	L CSEG	0054H 55 69#

ASSEMBLY COMPLETE, NO ERRORS FOUND

203830-29

intel

AP-70

NOTE

14th



APPLICATION NOTE

AP-252

Designing With The 80C51BH

TOM WILLIAMSON
MCO APPLICATIONS ENGINEER

September 1987

Order Number: 270068-002

CMOS EVOLVES

The original CMOS logic families were the 4000-series and the 74C-series circuits. The 74C-series circuits are functional equivalents to the corresponding numbered 74-series TTL circuits, but have CMOS logic levels and retain the other well known characteristics of CMOS logic.

These characteristics are: low power consumption, high noise immunity, and slow speed. The low power consumption is inherent to the nature of the CMOS circuit. The noise immunity is due partly to the CMOS logic levels, and partly to the slowness of the circuits. The slow speed is due to the technology used to construct the transistors in the circuit.

The technology used is called metal-gate CMOS, because the transistor gates are formed by metal deposition. More importantly, the gates are formed after the drain and source regions have been defined, and must overlap the source and drain somewhat to allow for alignment tolerances. This overlap plus the relatively large size of the transistors themselves result in high electrode capacitance, and that is what limits the speed of the circuit.

High speed CMOS became feasible with the development of the self-aligning silicon gate technology. In this process polysilicon gates are deposited before the source and drain regions are defined. Then the source and drain regions are formed by ion implantation using the gate itself as a mask for the implantation. This eliminates most of the overlap capacitance. In addition, the process allows smaller transistors. The result is a significant increase in circuit speed. The 74HC-series of CMOS logic circuits is based on this technology, and has speeds comparable to LS TTL, which is to say about 10 times faster than the 74C-series circuits.

The size reduction that contributes to the higher speed also demands an accompanying reduction in the maximum supply voltage. High-speed CMOS is generally limited to 6V.

WHAT IS CHMOS?

CHMOS is the name given to Intel's high-speed CMOS processes. There are two CHMOS processes, one based on an n-well structure and one based on a p-well structure. In the n-well structure, n-type wells are diffused into a p-type substrate. Then the n-channel transistors (nFETs) are built into the substrate and pFETs are built into the n-wells. In the p-well structure, p-type wells are diffused into an n-type substrate. Then the nFETs are built into the wells and pFETs, into the

substrate. Both processes have their advantages and disadvantages, which are largely transparent to the user.

Lower operating voltages are easier to obtain with the p-well structure than with the n-well structure. But the p-well structure does not easily adapt to an EPROM which would be pin-for-pin compatible with HMOS EPROMs. On the other hand the n-well structure can be based on the solidly founded HMOS process, in which nFETs are built into a p-type substrate. This allows somewhat more than half of the transistors in a CHMOS chip to be constructed by processes that are already well characterized.

Currently Intel's CHMOS microcontrollers and memory products are n-well devices, whereas CHMOS microprocessors are p-well devices.

Further discussion of the CHMOS technology is provided in References 1 and 2 (which are reprinted in the Microcontroller Handbook).

THE MCS®-51 FAMILY IN CHMOS

The 80C51BH is the CHMOS version of Intel's original 8051. The 80C31BH is the ROMless 80C51BH, equivalent to the 8031. These CHMOS devices are architecturally identical with their HMOS counterparts, except that they have two added features for reduced power. These are the Idle and Power Down modes of operation.

In most cases, an 80C51BH can directly replace the 8051 in existing applications. It can execute the same code at the same speed, accept signals from the same sources, and drive the same loads. However, the 80C51BH covers a wider range of speeds, will emit CMOS logic levels to CMOS loads, and will draw about 1/10 the current of an 8051 (and less yet in the reduced power modes). Interchangeability between the HMOS and CHMOS devices is discussed in more detail in the final section of this Application Note.

It should be noted that the 80C51BH CPU is not static. That means if the clock frequency is too low, the CPU might forget what it was doing. This is because the circuitry uses a number of dynamic nodes. A dynamic node is one that uses the node-to-ground capacitance to form a temporary storage cell. Dynamic nodes are used to reduce the transistor count, and hence the chip area, thus to produce a more economical device.

This is not to say that the on-chip RAM in CHMOS microcontrollers is dynamic. It's not. It's the CPU that is dynamic, and that is what imposes the minimum clock frequency specification.

Latchup is an SCR-type turn-on phenomenon that is the traditional nemesis of CMOS systems. The substrate, the wells, and the transistors form parasitic pnpn structures within the device. These parasitic structures turn on like an SCR if a sufficient amount of forward current is driven through one of the junctions. From the circuit designer's point of view it can happen whenever an input or output pin is externally driven a diode drop above V_{CC} or below V_{SS} , by a source that is capable of supplying the required trigger current.

However much of a problem latchup has been in the past, it is good to know that in most recently developed CMOS devices, and specifically in CHMOS devices, the current required to trigger latchup is typically well over 100 mA. The 80C51BH is virtually immune to latchup. (References 1 and 2 present a discussion of the latchup mechanisms and the steps that are taken on the chip to guard against it.) Modern CMOS is not absolutely immune to latchup, but with trigger currents in the hundreds of mA, latchup is certainly a lot easier to avoid than it once was.

A careless power-up sequence might trigger a latchup in the older CMOS families, but it's unlikely to be a major problem in high-speed CMOS or in CHMOS. There is still some risk incurred in inserting or removing chips or boards in a CMOS system while the power is on. Also, severe transients, such as inductive kicks or momentary short-circuits, can exceed the trigger current for latchup.

For applications in which some latchup risk seems unavoidable, you can put a small resistor (100 Ω or so) in series with signal lines to ensure that the trigger current will never be reached. This also helps to control overshoot and RFI.

LOGIC LEVELS AND INTERFACING PROBLEMS

CMOS logic levels differ from TTL levels in two ways.

Logic State	$V_{CC} = 5V$				
	74HC	74HCT	LS TTL	8051	80C51BH
V_{IH}	3.5V	2.0V	2.0V	2.0V	1.9V
V_{IL}	1.0V	0.8V	0.8V	0.8V	0.9V
V_{OH}	4.9V	4.9V	2.7V	2.4V	4.5V
V_{OL}	0.1V	0.1V	0.5V	0.45V	0.45V

Figure 1. Logic Level Comparison. (Output voltage levels depend on load current. Data sheets list guaranteed output levels for several load currents. The output levels listed here are for minimum loading.)

quires) a higher "logic 1" level than TTL. Secondly, CMOS logic levels are V_{CC} (or V_{DD}) dependent, whereas guaranteed TTL logic levels are fixed when V_{CC} is within TTL specs.

Standard 74HC logic levels are as follows:

$$\begin{aligned} V_{IHMIN} &= 70\% \text{ of } V_{CC} \\ V_{ILMAX} &= 20\% \text{ of } V_{CC} \\ V_{OHMIN} &= V_{CC} - 0.1V, |I_{OH}| \leq 20 \mu A \\ V_{OLMAX} &= 0.1V, |I_{OL}| \leq 20 \mu A \end{aligned}$$

Figure 1 compares 74HC, LS TTL, and 74HCT logic levels with those of the HMOS 8051 and the CHMOS 80C51BH for $V_{CC} = 5V$.

Output logic levels depend of course on load current, and are normally specified at several load currents. When CMOS and TTL are powered by the same V_{CC} , the logic levels guaranteed on the data sheets indicate that CMOS can drive TTL, but TTL can't drive CMOS. The incompatibility is that the TTL circuit's V_{OH} level is too low to reliably be recognized by the CMOS circuit as a valid V_{IH} .

Since HMOS circuits were designed to be TTL-compatible, they have the same incompatibility.

Fortunately, 74HCT-series circuits are available to ease these interfacing problems. They have TTL-compatible logic levels at the inputs and standard CMOS levels at the outputs.

The 80C51BH is designed to work with either TTL or CMOS. Therefore its logic levels are specified very much like 74HCT circuits. That is, its input logic levels are TTL-compatible, and its output characteristics are like standard high-speed CMOS.

NOISE CONSIDERATIONS

One of the major reasons for going to CMOS has traditionally been that CMOS is less susceptible to noise. As previously noted, its low susceptibility to noise is

partly due to superior noise margins, and partly due to its slow speed.

Noise margin is the difference between V_{OL} and V_{IL} , or between V_{OH} and V_{IH} . If V_{OH} from a driving circuit is 2.7V and V_{IH} to the driven circuit is 2.0V, then the driven circuit has 0.7V of noise margin at the logic high level. These kinds of comparisons show that an all-CMOS system has wider noise margins than an all-TTL system.

Figure 2 shows noise margins in CMOS and LS TTL systems when both have $V_{CC} = 5V$. It can be seen that CMOS/CMOS and CMOS/CHMOS systems have an edge over LS TTL in this respect.

Noise margins can be misleading, however, because they don't say how much noise energy it takes to induce in the circuit a noise voltage of sufficient amplitude to cause a logic error. This would involve consideration of the width of the noise pulse as compared with the circuit's response speed, and the impedance to ground from the point of noise introduction in the circuit.

When these considerations are included, it is seen that using the slower 74C- and 4000-series circuits with a 12 or 15V supply voltage does offer a truly improved level of noise immunity, but that high-speed CMOS at 5V is not significantly better than TTL.

One should not mistake the wider supply voltage tolerance of high-speed CMOS for V_{CC} glitch immunity. Supply voltage tolerance is a DC rating, not a glitch rating.

For any clocked CMOS, and most especially for VLSI CMOS, V_{CC} decoupling is critical. CHMOS draws

Interface	Noise Margin for $V_{CC} = 5V$	
	Logic Low $V_{IL} - V_{OL}$	Logic High $V_{OH} - V_{IH}$
74HC to 74HC	0.9V	1.4V
LSTTL to LSTTL	0.3V	0.7V
LSTTL to 74HCT	0.3V	0.7V
LSTTL to 80C51BH	0.3V	0.7V
74HC to 80C51BH	0.8V	3.0V
80C51BH to 74HC	0.8V	1.0V

Figure 2. Noise Margins for CMOS and LS TTL Circuits

current in extremely sharp spikes at the clock edges. The VHF and UHF components of these spikes are not drawn from the power supply, but from the decoupling capacitor. If the decoupling circuit is not sufficiently low in inductance, V_{CC} will glitch at each clock edge. We suggest that a 0.1 μF decoupler cap be used in a minimum-inductance configuration with the microcontroller. A minimum-inductance configuration is one that minimizes the area of the loop formed by the chip (V_{CC} to V_{SS}), the traces to the decoupler cap, and the decoupler cap. PCB designers too often fail to understand that if the traces that connect the decoupler cap to the V_{CC} and V_{SS} pins aren't short and direct, the decoupler loses much of its effectiveness.

Overshoot and ringing in signal lines are potential sources of logic upsets. These can largely be controlled by circuit layout. Inserting small resistors (about 100 Ω) in series with signal lines that seem to need them will also help.

The sharp edges produced by high-speed CMOS can cause RFI problems. The severity of these problems is largely a function of the PCB layout. We don't mean to imply that all RFI problems can be solved by a better PCB layout. It may well be, for example, that in some RFI-sensitive designs high-speed CMOS is simply not the answer. But circuit layout is a critical factor in the noise performance of any electronic system, and more so in high-speed CMOS systems than others.

Circuit layout techniques for minimizing noise susceptibility and generation are discussed in References 3 through 6.

UNUSED PINS

CMOS input pins should not be left to float, but should always be pulled to one logic level or the other. If they float, they tend to float into the transition region between 0 and 1, where the pullup and pulldown devices in the input buffer are both conductive. This causes a significant increase in I_{CC} . A similar effect exists in HMOS circuits, but with less noticeable results.

In 80C51BH and 80C31BH designs, unused pins of Ports 1, 2, and 3 can be ignored, because they have internal pullups that will hold them at a valid Logic 1 level. Port 0 pins are different, however, in not having internal pullups (except during bus operations).

When the 80C51BH is in reset, the Port 0 pins are in a float state unless they are externally pulled up or down. If it's going to be held in reset for just a short time, the transient float state can probably be ignored. When it comes out of reset, the pins stay afloat unless

they are externally pulled either up or down. Alternatively, the software can internally write 0s to whatever Port 0 pins may be unused.

The same considerations are applicable to the 80C31BH with regards to reset. But when the 80C31BH comes out of reset, it commences bus operations, during which the logic levels at the pins are always well defined as high or low.

Consider the 80C31BH in the Power Down or Idle modes, however. In those modes it is not fetching instructions, and the Port 0 pins will float if not externally pulled high or low. The choice of whether to pull them high or low is the designer's. Normally it is sufficient to pull them up to V_{CC} with 10k resistors. But if power is going to be removed from circuits that are connected to the bus, it will be advisable to pull the bus pins down (normally with 10k resistors). Considerations involved in selecting pullup and pulldown resistor values are as follows.

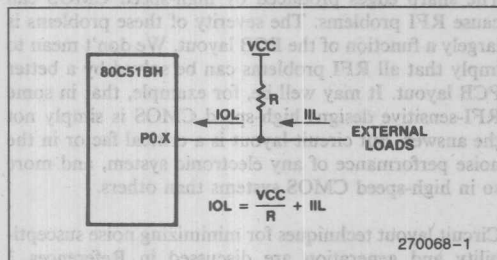


Figure 3a. Conditions defining the minimum value for R. P0.X is emitting a logic low. R must be large enough to not cause IOL to exceed data sheet specifications.

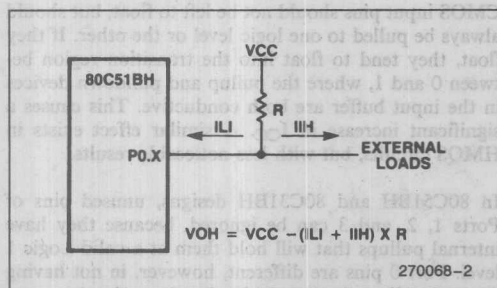


Figure 3b. Conditions defining the maximum value for R. P0.X is in a high impedance state. R must be small enough to keep V_{OH} acceptably high.

PULLUP RESISTORS

If a pullup resistor is to be used on a Port 0 pin, its minimum value is determined by I_{OL} requirements. If the pin is trying to emit a 0, then it will have to sink the current from the pullup resistor plus whatever other current may be sourced by other loads connected to the pin, as shown in Figure 3a, while maintaining a valid output low (V_{OL}). To guarantee that the pin voltage will not exceed 0.45V, the resistor should be selected so that I_{OL} doesn't exceed the value specified on the data sheet. In most CMOS applications, the minimum value would be about 2k Ω .

The maximum value you could use depends on how fast you want the pin to pull up after bus operations have ceased, and how high you want the V_{OH} level to be. The smaller the resistor the faster it pulls up. Its effect on the V_{OH} level is that $VOH = VCC - (ILI + IIH) \times R$. ILI is the input leakage current to the Port 0 pin, and IIH is the input high current to the external loads, as shown in Figure 3b. Normally V_{OH} can be expected to reach 0.9 V_{CC} if the pullup resistance does not exceed about 50k Ω .

Pulldown Resistors

If a pulldown resistor is to be used on a Port 0 pin, its minimum value is determined by V_{OH} requirements during bus operations, and its maximum value is in most cases determined by leakage current.

During bus operations the port uses internal pullups to emit 1s. The D.C. Characteristics in the data sheet list guaranteed V_{OH} levels for given I_{OH} currents. (The "-" sign in the I_{OH} value means the pin is sourcing that current to the external load, as shown in Figure 4.) To ensure the V_{OH} level listed in the data sheet, the resis-

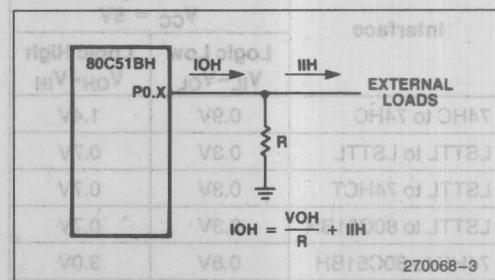


Figure 4a. Conditions defining the minimum value for R. P0.X is emitting a 1 in a bus operation. R must be large enough to not cause I_{OH} to exceed data sheet specifications.

CMOS circuits draw current in two directions during logic transitions. The average current is made up of two components. One is the current that flows during the transition when the input signal is changing. The other is the current that flows during the steady-state high or low state. The average current is larger when the transition time of the input signal is shorter.

When the pin goes into a high impedance state, the pulldown resistor will have to sink leakage current from the pin, plus whatever other current may be sourced by other loads connected to the pin, as shown in Figure 4b. The Port 0 leakage current is I_{LI} on the data sheet. The resistor should be selected so that the voltage developed across it by these currents will be seen as a logic low by whatever circuits are connected to it (including the 80C51BH). In CMOS/CHMOS applications, 50k Ω is normally a reasonable maximum value.

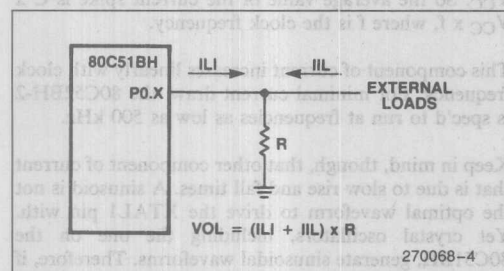


Figure 4b. Conditions defining the maximum value for R. P0.X is in a high impedance state. R must be small enough to keep VOL acceptably low.

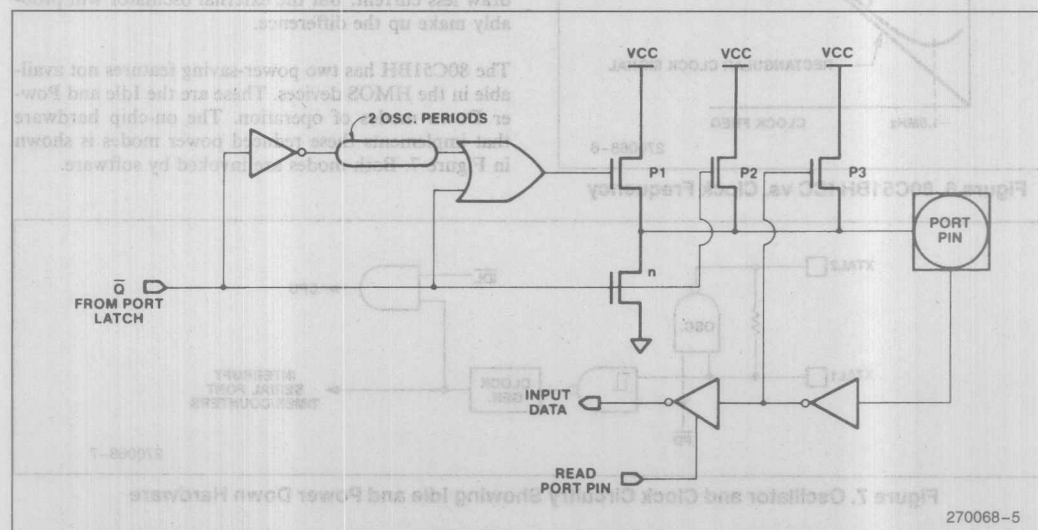


Figure 5. 80C51BH Output Drivers for Ports 1, 2 and 3

DRIVE CAPABILITY OF THE INTERNAL PULLUPS

There's an important difference between HMOS and CHMOS port drivers. The pins of Ports 1, 2, and 3 of the CHMOS parts each have three pullups: strong, normal, and weak, as shown in Figure 5. The strong pullup (p1) is only used during 0-to-1 transitions, to hasten the transition. The weak pullup (p2) is on whenever the bit latch contains a 1. The "normal" pullup (p3) is controlled by the pin voltage itself.

The reason that p3 is controlled by the pin voltage is that if the pin is being used as an input, and the external source pulls it to a low, then turning off p3 makes for a lower I_{IL} . The data sheet shows an " I_{IL} " specification. This is the current that p3 will source during the time the pin voltage is making its 1-to-0 transition. This is what I_{IL} would be if an input low at the pin didn't turn p3 off.

Note, however, that this p3 turn-off mechanism puts a restriction on the drive capacity of the pin if it's being used as an output. If you're trying to output a logic high, and the external load pulls the pin voltage below the pin's V_{IHMIN} spec, p3 might turn off, leaving only the weak p2 to provide drive to the load. To prevent this happening, you need to ensure that the load doesn't draw more than the I_{OH} spec for a valid V_{OH} . The idea is to make sure the pin voltage never falls below its own V_{IHMIN} specification.

The main reason for going to CMOS, of course, is to conserve power. (There are other reasons, but this is the main one.) Conserving power doesn't mean just reducing your electric bill. Nor does it necessarily relate to battery operation, although battery operation without CMOS is pretty unhandy. The main reason for conserving power is to be able to put more functionality into a smaller space. The reduced power consumption allows the use of smaller and lighter power supplies, and less heat being generated allows denser packaging of circuit components. Expensive fans and blowers can usually be eliminated.

A cooler running chip is also more reliable, since most random and wearout failures relate to die temperature. And finally, the lower power dissipation will allow more functions to be integrated onto the chip.

The reason CMOS consumes less power than NMOS is that when it's in a stable state there is no path of conduction from V_{CC} to V_{SS} except through various leakage paths. CMOS does draw current when it's changing states. How much current it draws depends on how often and how quickly it changes states.

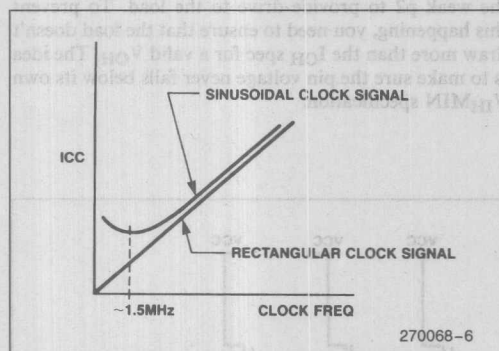


Figure 6. 80C51BH ICC vs. Clock Frequency

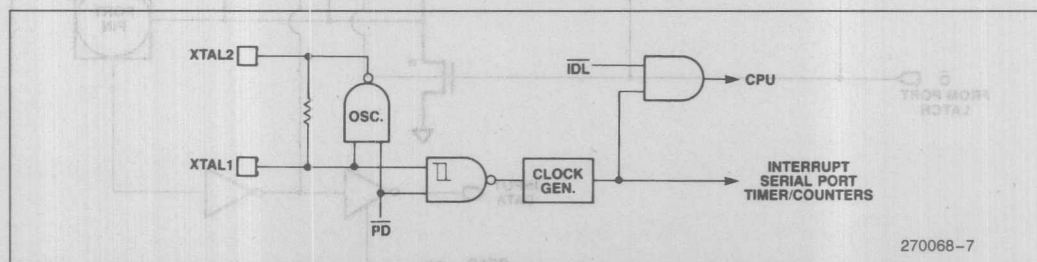


Figure 7. Oscillator and Clock Circuitry Showing Idle and Power Down Hardware

ical transitions. These current spikes are made up of two components. One is the current that flows during the transition time when pullup and pulldown FETs are both active. The average (DC) value of this component is larger when the transition times of the input signals are longer. For this reason, if the current draw is a critical factor in the design, slow rise and fall times should be avoided, even when the system speed doesn't seem to justify a need for nanosecond switching speeds.

The other component is the current that charges stray and load capacitance at the nodes of a CMOS logic gate. The average value of this current spike is its area (integral over time) multiplied by its rep rate. Its area is the amount of charge it takes to raise the node capacitance, C , to V_{CC} . That amount of charge is just $C \times V_{CC}$. So the average value of the current spike is $C \times V_{CC} \times f$, where f is the clock frequency.

This component of current increases linearly with clock frequency. For minimal current draw, the 80C52BH-2 is spec'd to run at frequencies as low as 500 kHz.

Keep in mind, though, that other component of current that is due to slow rise and fall times. A sinusoid is not the optimal waveform to drive the XTAL1 pin with. Yet crystal oscillators, including the one on the 80C51BH, generate sinusoidal waveforms. Therefore, if the on-chip oscillator is being used, you can expect the device to draw more current at 500 kHz, than it does at 1.5 MHz, as shown in Figure 6. If you derive a good sharp square wave from an external oscillator, and use that to drive XTAL1, then the microcontroller will draw less current. But the external oscillator will probably make up the difference.

The 80C51BH has two power-saving features not available in the HMOS devices. These are the Idle and Power Down modes of operation. The on-chip hardware that implements these reduced power modes is shown in Figure 7. Both modes are invoked by software.

Idle: In the Idle Mode ($\overline{\text{IDL}} = 0$ in Figure 7), the CPU puts itself to sleep by gating off its own clock. It doesn't stop the oscillator. It just stops the internal clock signal from getting to the CPU. Since the CPU draws 80 to 90 percent of the chip's power, shutting it off represents a fairly significant power savings. The on-chip peripherals (timers, serial port, interrupts, etc.) and RAM continue to function as normal. The CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle.

The Idle Mode is invoked by setting bit 0 (IDL) of the PCON register. PCON is not bit-addressable, so the bit has to be set by a byte operation, such as

```
ORL PCON, #1
```

The PCON register also contains flag bits GF0 and GF1, which can be used for any general purposes, or to give an indication if an interrupt occurred during normal operation or during Idle. In this application, the instruction that invokes Idle also sets one or both of the flag bits. Their status can then be checked in the interrupt routines.

While the device is in the Idle Mode, ALE and PSEN emit logic high (V_{OH}), as shown in Figure 8. This is so external EPROM can be deselected and have its output disabled.

The port pins hold the logical states they had at the time the Idle was activated. If the device was executing out of external program memory, Port 0 is left in a high impedance state and Port 2 continues to emit the high byte of the program counter (using the strong pullups to emit 1s). If the device was executing out of internal program memory, Ports 0 and 2 continue to emit whatever is in the P0 and P2 registers.

There are two ways to terminate Idle. Activation of any enabled interrupt will cause the hardware to clear bit 0 of the PCON register, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that invoked Idle.

The other way is with a hardware reset. Since the clock oscillator is still running, RST only needs to be held active for two machine cycles (24 oscillator periods) to complete the reset. Note that this exit from Idle writes 1s to all the ports, initializes all SFRs to their reset values, and restarts program execution from location 0.

Power Down: In the Power Down Mode ($\overline{\text{PD}} = 0$ in Figure 7), the CPU puts the whole chip to sleep by turning off the oscillator. In case it was running from an external oscillator, it also gates off the path to the internal phase generators, so no internal clock is generated even if the external oscillator is still running. The on-chip RAM, however, saves its data, as long as V_{CC} is maintained. In this mode the only I_{CC} that flows is leakage, which is normally in the micro-amp range.

The Power Down Mode is invoked by setting bit 1 in the PCON register, using a byte instruction such as

```
ORL PCON, #2
```

While the device is in Power Down, ALE and PSEN emit lows (V_{OL}), as shown in Figure 8. The reason they are designed to emit lows is so that power can be removed from the rest of the circuit, if desired, while the 80CS51BH is in its Power Down mode.

The port pins continue to emit whatever data was written to them. Note that Port 2 emits its P2 register data even if execution was from external program memory.

Pin	Internal Execution		External Execution	
	Idle	Power Down	Idle	Power Down
ALE	1	0	1	0
PSEN/	1	0	1	0
P0	SFR Data	SFR Data	High-Z	High-Z
P1	SFR Data	SFR Data	SFR Data	SFR Data
P2	SFR Data	SFR Data	PCH	SFR Data
P3	SFR Data	SFR Data	SFR Data	SFR Data

Figure 8. Status of Pins in Idle and Power Down Modes. "SFR data" means the port pins emit their internal register data. "PCH" is the high byte of the Program Counter.

In Figure 9, when V_{CC} is switched on to the 80C31BH, capacitor C1 provides a power-on reset. The reset function writes 1s to all the port pins. The 1 at P2.6 turns Q1 on, enabling V_{CC} to the secondary circuit through transistor Q2. As the 80C31BH comes out of reset, Port 2 commences emitting the high byte of the Program Counter, which results in the P2.7 and P2.6 pins outputting 0s. The 0 at P2.7 ensures continuation of V_{CC} to the secondary circuit.

The system software must now write a 1 to P2.7 and a 0 to P2.6 in the Port 2 SFR, P2. These values will not appear at the Port 2 pins as long as the device is fetching instructions from external program memory. However, whenever the 80C31BH goes into Power Down, these values will appear at the port pins, and will shut off both transistors, disabling V_{CC} to the secondary circuit.

Closing the switch S1 re-energizes the secondary circuit, and at the same time sends a reset through C2 to the 80C31BH to wake it up. The diode D1 is to prevent C1 from hogging current from C2 during this secondary reset. D2 prevents C2 from discharging through the RST pin when V_{CC} to the secondary circuit goes to zero.

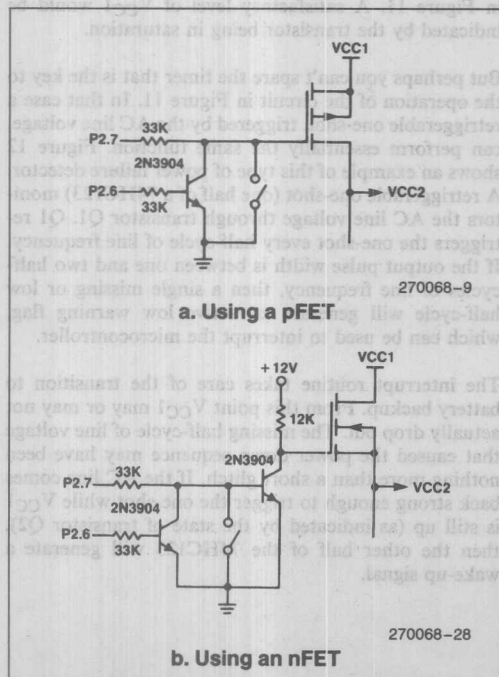


Figure 10. Using Power MOSFETs to Control V_{CC2}

USING POWER MOSFETs to CONTROL V_{CC}

Power MOSFETs are gaining in popularity (and availability). The easiest way to control V_{CC} is with a Logic Level pFET, as shown in Figure 10a. This circuit allows the full V_{CC} to be used to turn the device on. Unfortunately, power pFETs are not economically competitive with bipolar transistors of comparable ratings.

Power nFETs are both economical and available, and can be used in this application if a DC supply of higher voltage is available to drive the gate. Figure 10b shows how to implement a V_{CC} switch using a power nFET and a (nominally) +12V supply. The problem here is that if the device is on, its source voltage is +5V. To maintain the on state, the gate has to be another 5 or 10V above that. The "12V" supply is not particularly critical. A minimally filtered, unregulated rectifier will suffice.

BATTERY BACKUP SYSTEMS

Here we consider circuits that normally draw power from the AC line, but switch to battery operation in the event of a power failure. We assume that in battery operation high-current loads will be allowed to die along with the AC power. The system may continue then with reduced functionality, monitoring a control transducer, perhaps, or driving an LCD. Or it may go into a bare-bones survival mode, in which critical data is saved but nothing else happens till AC power is restored.

In any case it is necessary to have some early warning of an impending power failure so that the system can arrange an orderly transfer to battery power. Early warning systems can operate by monitoring either the AC line voltage or the unregulated rectifier output, or even by monitoring the regulated DC voltage.

Monitoring the AC line voltage gives the earliest warning. That way you can know within one or two half-cycles of line frequency that AC power is down. In most cases you then have at least another half-cycle of line frequency before the regulated V_{CC} starts to fall. In a half-cycle of line frequency an 80C51BH can execute about 5,000 instructions—plenty of time to arrange an orderly transfer of power.

The circuit in Figure 11 uses a Zener diode to test the line voltage each half-cycle, and a junction transistor to pass the information on to the 80C51BH. (Obviously a voltage comparator with a suitable reference source can

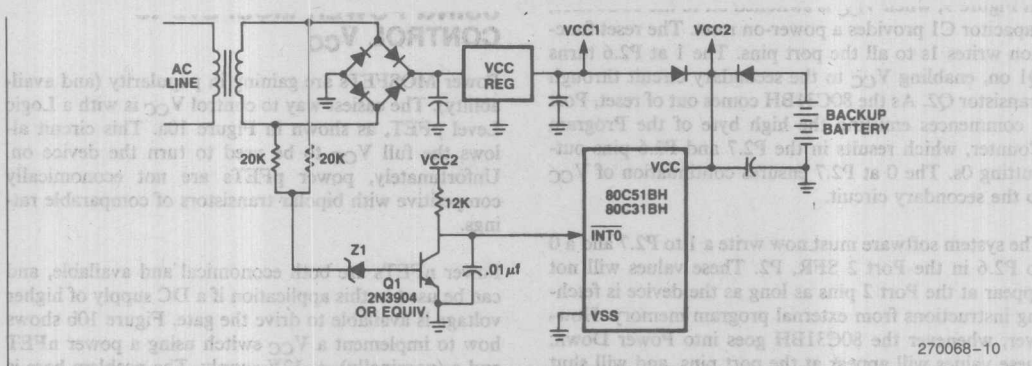


Figure 11. Power Failure Detector with Battery Backup. When AC power fails, VCC1 goes down and VCC2 is held.

perform the same function, if one prefers.) The way it works is if the line voltage reaches an acceptably high level, it breaks over Z1, drives Q1 to saturation, and interrupts the 80C51BH. The interrupt would be transition-activated, in this application. The interrupt service routine reloads one of the C51BH's timers to a value that will make it roll over in something between one and two half-cycles of line frequency. As long as the line voltage is healthy, the timer never rolls over, because it is reloaded every half-cycle. If there is a single half-cycle in which the line voltage doesn't reach a high enough level to generate the interrupt, the timer rolls over and generates a timer interrupt.

The timer interrupt then commences the transition to battery backup. Critical data needs to be copied into protected RAM. Signals to circuits that are going to lose power must be written to logic low. Protected circuits (those powered by VCC2) that communicate with unprotected circuits must be deselected. The microcontroller itself may be put into Idle, so that it can continue some level of interrupt-driven functionality, or it may be put into Power Down.

Note that if the CPU is going to invoke Power Down, the Special Function Registers may also need to be copied into protected RAM, since the reset that terminates the Power Down mode will also initialize all the SFRs to their reset values.

The circuit in Figure 11 does not show a wake-up mechanism. A number of choices are available, however. A pushbutton could be used to generate an interrupt, if the CPU is in Idle, or to activate reset, if the CPU is in Power Down.

Automatic wake-up on power restoration is also possible. If the CPU is in Idle, it can continue to respond to any interrupts that might be generated by Q1. The interrupt service routine determines from the status of flag bits GF0 and GF1 in PCON that it is in Idle because there was a power outage. It can then sample VCC1 through a voltage comparator similar to Z1, Q1 in Figure 11. A satisfactory level of VCC1 would be indicated by the transistor being in saturation.

But perhaps you can't spare the timer that is the key to the operation of the circuit in Figure 11. In that case a retriggerable one-shot, triggered by the AC line voltage, can perform essentially the same function. Figure 12 shows an example of this type of power failure detector. A retriggerable one-shot (one half of a 74HC123) monitors the AC line voltage through transistor Q1. Q1 re-triggers the one-shot every half-cycle of line frequency. If the output pulse width is between one and two half-cycles of line frequency, then a single missing or low half-cycle will generate an active low warning flag, which can be used to interrupt the microcontroller.

The interrupt routine takes care of the transition to battery backup. From this point VCC1 may or may not actually drop out. The missing half-cycle of line voltage that caused the power down sequence may have been nothing more than a short glitch. If the AC line comes back strong enough to trigger the one-shot while VCC1 is still up (as indicated by the state of transistor Q2), then the other half of the 74HC123 will generate a wake-up signal.

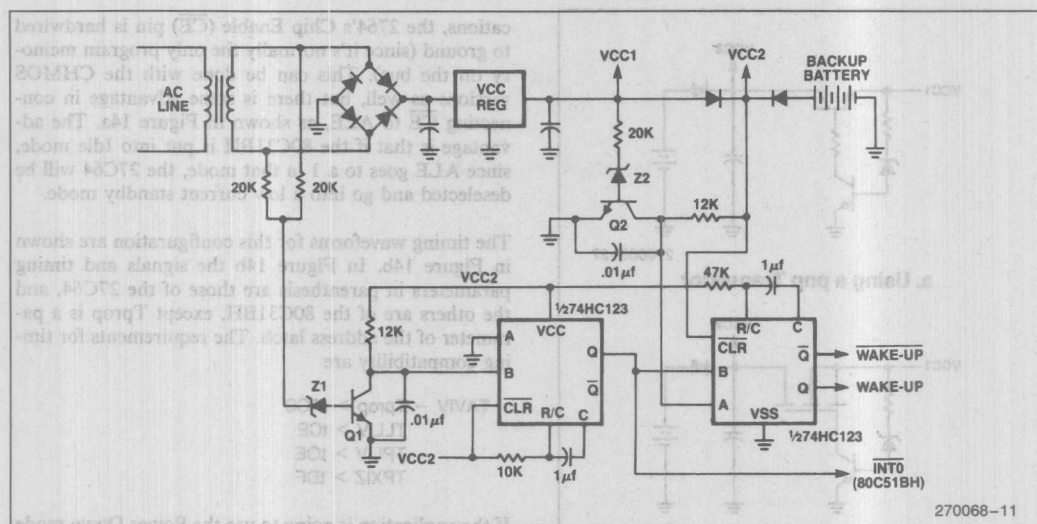


Figure 12. Power Failure Detector uses retriggerable one-shots to flag impending power outage and generate automatic wake-up when power returns.

Having been awakened, the 80C51BH will stay awake for at least another half-cycle of line frequency (another 5,000 or so instructions) before possibly being told to arrange another transfer of power. Consequently, if the line voltage is jittering erratically around the switchover point (determined by diode Z1), the system will limp along executing in half-cycle units of line frequency.

On the other hand, if the power outage is real and lengthy, V_{CC1} will eventually fall below the level at which the backup battery takes over. The backup battery maintains power to the 80C51BH, and to the 74HC123, and to whatever other circuits are being protected during this outage. The battery voltage must be high enough to maintain V_{CCMIN} specs to the 80C51BH.

If the microcontroller is an 80C31BH, executing out of external ROM, and if the C31BH is put into Idle during the power outage, then the external ROM must also be supplied by the battery. On the other hand, if the C31BH is put into Power Down during the outage, then the ROM can be allowed to die with the AC power. The considerations here are the same as in Figure 9: V_{CC} to the ROM is still up at the time Power Down is invoked, and we must ensure (through selection of diode Z2 in Figure 12) that the 80C31BH is not awakened till ROM power is back in spec.

POWER SWITCHOVER CIRCUITS

Battery backup systems need to have a way for the protected circuits to draw power from the line-operated power supply when that source is available, and to switch over to battery power when required. The switchover circuit is simple if the entire system is to be battery powered in the event of a line power outage. In that case a pair of diodes suffice, as shown in Figure 12, provided V_{CCMIN} specs are still met after the diode drop has been subtracted from its respective power source.

The situation becomes more complicated when part of the circuit is going to be allowed to die when the AC power goes out. In that case it is difficult to maintain equal V_{CC} s to protected and unprotected circuits (and possibly dangerous not to).

The problem can be alleviated by using a Schottky diode instead of a 1N4001, for its lower forward voltage drop. The 1N5820, for example, has a forward drop of about 0.35V at 1A.

Other solutions are to use a transistor or power MOSFET switch, as shown in Figure 13. With minor modifications this switch can be controlled by port pins.

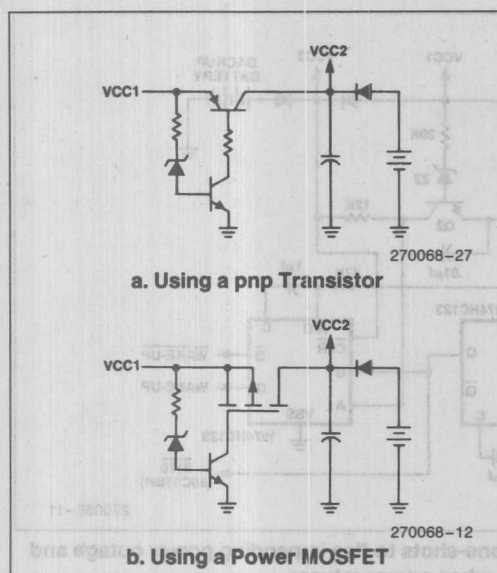


Figure 13. Power Switchover Ckts.

80C31BH + CHMOS EPROM

The 27C64 and 87C64 are Intel's 8K byte CHMOS EPROMs. The 27C64 requires an external address latch, and can be used with the 80C31BH as shown in Figure 14a. In most 8031 + 2764 (HMOS) appli-

cations, the 2764's Chip Enable ($\overline{\text{CE}}$) pin is hardwired to ground (since it's normally the only program memory on the bus). This can be done with the CHMOS versions as well, but there is some advantage in connecting $\overline{\text{CE}}$ to ALE, as shown in Figure 14a. The advantage is that if the 80C31BH is put into Idle mode, since ALE goes to a 1 in that mode, the 27C64 will be deselected and go into a low current standby mode.

The timing waveforms for this configuration are shown in Figure 14b. In Figure 14b the signals and timing parameters in parenthesis are those of the 27C64, and the others are of the 80C31BH, except Tprop is a parameter of the address latch. The requirements for timing compatibility are

TAVIV – Tprop > tACO
TLLIV > tCE
TPLIV > tOE
TPXIZ > tDF

If the application is going to use the Power Down mode then we have another consideration: In Idle, $\overline{\text{ALE}} = \overline{\text{PSEN}} = 1$, and in Power Down, $\overline{\text{ALE}} = \overline{\text{PSEN}} = 0$. In a realistic application there are likely to be more chips in the circuit than are shown in Figure 14, and it is likely that the nonessential ones will have their V_{CC} removed while the CPU is in Power Down. In that case the EPROM and the address latch should be among the chips that have V_{CC} removed, and logic lows are exactly what are required at $\overline{\text{ALE}}$ and $\overline{\text{PSEN}}$.

But if V_{CC} is going to be maintained to the EPROM during Power Down, then it will be necessary to de-

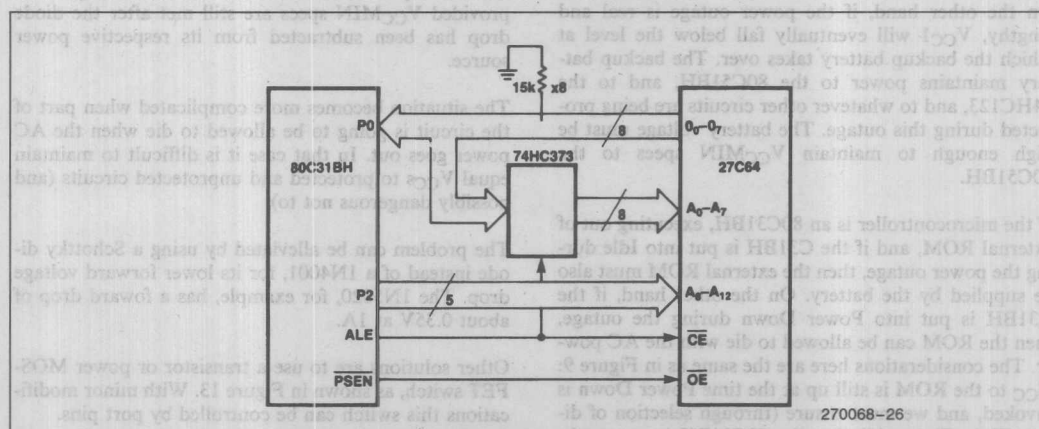


Figure 14a. 80C31BH + 27C64

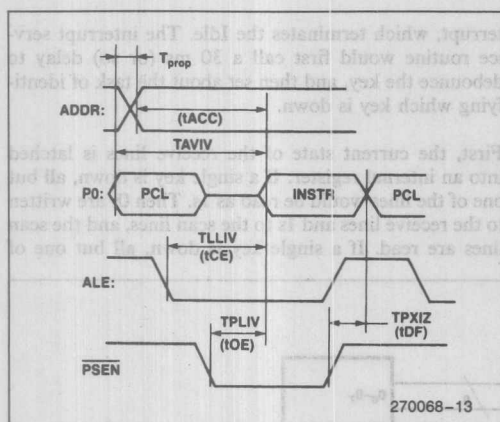


Figure 14b. Timing Waveforms for 80C31BH + 27C64

select the EPROM when the CPU is in Power Down. If Idle is never invoked, \overline{CE} of the EPROM can be connected to P2.7 of the 80C31BH, as shown in Figure 15a. In normal operation, P2.7 will be emitting the MSB of the Program Counter, which is 0 if the program contains less than 32K of code. Then when the CPU goes into Power Down, the Port 2 pins emit P2 SFR data, which puts a 1 at P2.7, thus deselecting the EPROM.

If Idle and Power Down are both going to be used, \overline{CE} of the EPROM can be driven by the logical OR of ALE and P2.7, as shown in Figure 15b. In Idle, ALE = 1 will deselect the EPROM, and in Power Down, P2.7 = 1 will deselect it.

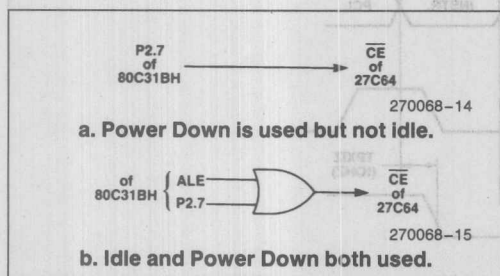


Figure 15. Modifications to 80C31BH/27C64 Interface

Pulldown resistors are shown in Figure 14a under the assumption that something on the bus is going to have its V_{CC} removed during Power Down. If this is not the case, pullups can be used as well as pulldowns.

The 87C64 is like the 27C64 except that it has an on-chip address latch. The Port 0 pins are tied to both address and data pins of the 87C64, as shown in Figure 16a. ALE drives the EPROM's ALE/ \overline{CS} input. During ALE high, the address information is allowed to flow into the EPROM and begin accessing the code byte. On the falling edge of ALE the address byte is internally latched. The A0-A7 inputs are then ignored and the same bus lines are used to transmit the fetched code byte from the 00-07 pins back to the 80C31BH.

The timing waveforms for this configuration are shown in Figure 16b. In Figure 16b the signals and timing parameters in parentheses are those of the 87C64, and the others are of the 80C31BH. The requirements for timing compatibility are

- TLHLL > tLL
- TAVLL > tAL
- TLLAX > tLA
- TLLIV > tACL
- TPLIV > tOE
- TLLPL > tCOE
- TPXIZ > tOHZ

The same considerations apply to the 87C64 as to the 27C64 with regards to the Idle and Power Down modes. Basically you want $\overline{CS} = 1$ if V_{CC} is maintained to the EPROM, and $\overline{CS} = \overline{OE} = 0$ if V_{CC} is removed.

SCANNING A KEYBOARD

There are many different kinds of keyboards, but alphanumeric keyboards generally consist of a matrix of 8 scan lines and 8 receive lines as shown in Figure 17. Each set of lines connects to one port of the microcontroller. The software has written 0s to the scan lines, and 1s to the receive lines. Pressing a key connects a scan line to a receive line, thus pulling the receive line to a logic low.

The 8 receive lines are ANDed to one of the external interrupt pins, so that pulling any of the receive lines low generates an interrupt. The interrupt service routine has to identify the pressed key, if only one key is down, and convert that information to some useful output. If more than one key in the line matrix is found to be pressed, no action is taken. (This is a "two key lock-out" scheme.)

On some keyboards, certain keys (Shift, Control, Escape, etc.) are not a part of the line matrix. These keys would connect directly to a port pin on the microcontroller, and would not cause lock-out if pressed simultaneously with a matrix key, nor generate an interrupt if pressed singly.

Normally the microcontroller would be in idle mode when a key has not been pressed, and another task is not in progress. Pressing a matrix key generates an in-

terrupt, which terminates the Idle. The interrupt service routine would first call a 30 ms (or so) delay to debounce the key, and then set about the task of identifying which key is down.

First, the current state of the receive lines is latched into an internal register. If a single key is down, all but one of the lines would be read as 1s. Then 0s are written to the receive lines and 1s to the scan lines, and the scan lines are read. If a single key is down, all but one of

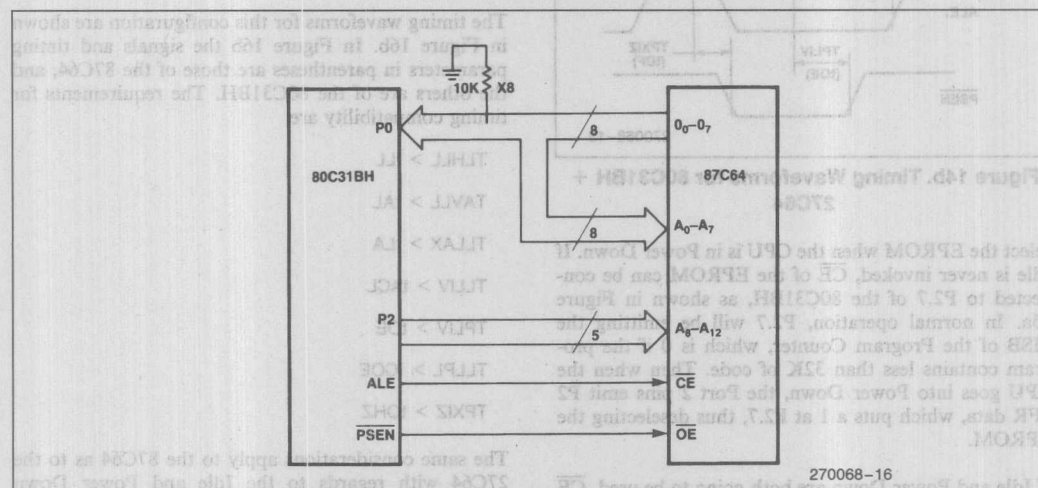


Figure 16a. 80C31BH + 87C64

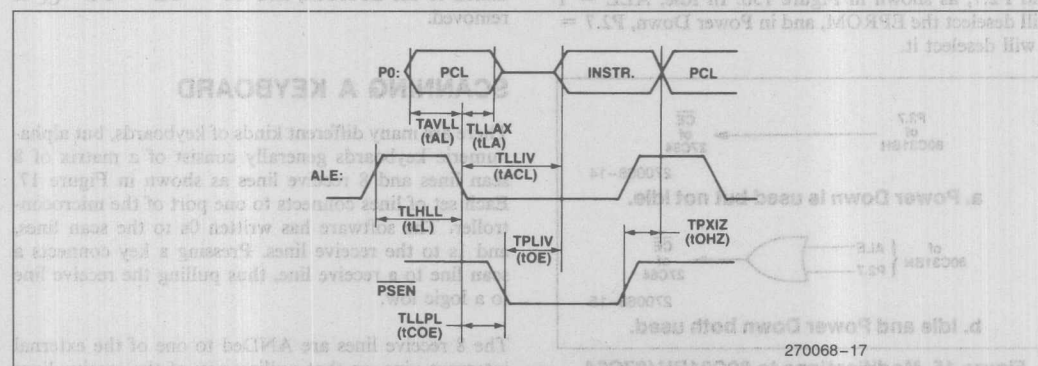


Figure 16b. Timing Waveforms for 80C31BH + 87C64

these lines would be read as 1s. By locating the single 0 in each set of lines, the pressed key can be identified. If more than one matrix key is down, one or both sets of lines will contain multiple 0s.

A subroutine is used to determine which of 8 bits in either set of lines is 0, and whether more than one bit is 0. Figure 18 shows a subroutine (SCAN) which does that using the 8051's bit-addressing capability. To use the subroutine, move the line data into a bit-addressable RAM location named LINE, and call the SCAN routine. The number of LINE bits which are zero is returned in ZERO_COUNTER. If only one bit is zero, its number (1 through 8) is returned in ZERO_BIT.

The interrupt service routine that is executed in response to a key closure might then be as follows:

RESPONSE_TO_KEY_CLOSURE:

```
CALL DEBOUNCE_DELAY
MOV LINE,P1; ;See Figure 17.
CALL SCAN
DJNZ ZERO_COUNTER,REJECT
MOV ADDRESS,ZERO_BIT
MOV P2,#OFFH; ;See Figure 17.
MOV P1,#0
MOV LINE,P2
CALL SCAN
DJNZ ZERO_COUNTER,REJECT
XCH A,ZERO_BIT
SWAP A
ORL ADDRESS,A
XCH A,ZERO_BIT
MOV P1,#OFFH
MOV P2,#0
REJECT: CLR EX0
RETI
```

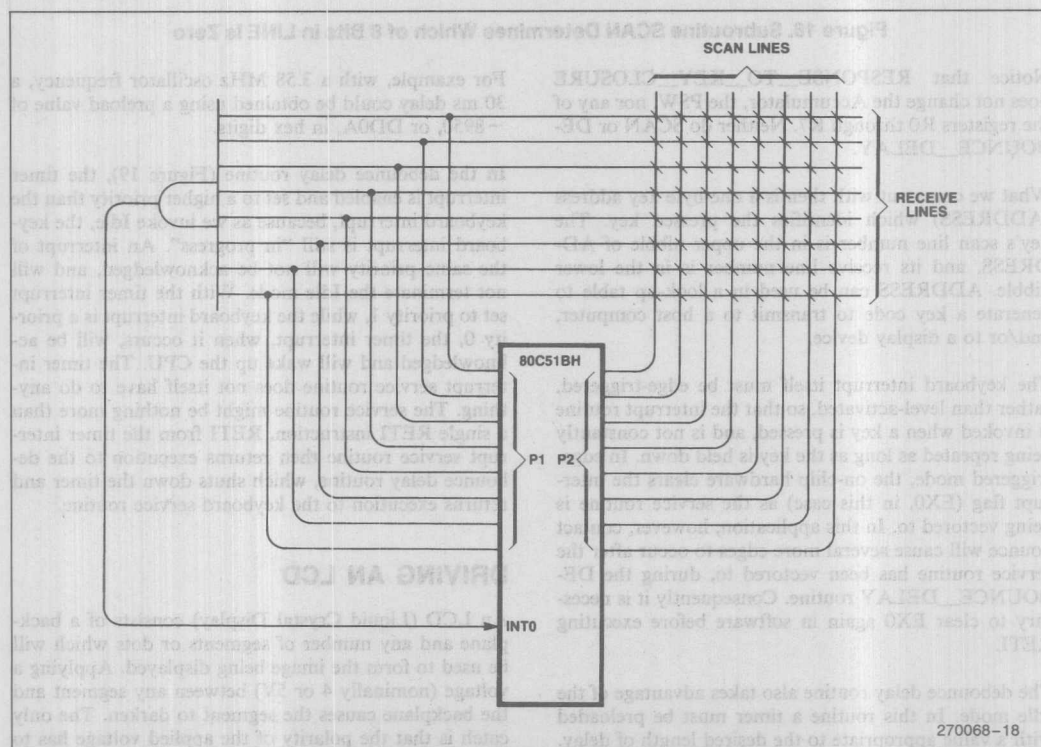


Figure 17. Scanning a Keyboard

SCAN:	MOV	ZERO_COUNTER,#0	: ZERO_COUNTER counts the number of 0s in LINE.
	JB	LINE_0,ONE	: Test LINE bit 0.
	INC	ZERO_COUNTER	: If LINE_0 = 0, increment ZERO_COUNTER
	MOV	ZERO_BIT,#1	: and record that line number 1 is active.
ONE:	JB	LINE_1,TWO	: Procedure continues for other LINE bits.
	INC	ZERO_COUNTER	
	MOV	ZERO_BIT,#2	: Line number 2 is active.
TWO:	JB	LINE_2,THREE	
	INC	ZERO_COUNTER	: Line number 3 is active.
	MOV	ZERO_BIT,#3	
THREE:	JB	LINE_3,FOUR	: Line number 4 is active.
	INC	ZERO_COUNTER	
	MOV	ZERO_BIT,#4	: Line number 5 is active.
FOUR:	JB	LINE_4,FIVE	
	INC	ZERO_COUNTER	: Line number 6 is active.
	MOV	ZERO_BIT,#5	
FIVE:	JB	LINE_5,SIX	: Line number 7 is active.
	INC	ZERO_COUNTER	
	MOV	ZERO_BIT,#6	: Line number 8 is active.
SIX:	JB	LINE_6,SEVEN	
	INC	ZERO_COUNTER	
	MOV	ZERO_BIT,#7	
SEVEN:	JB	LINE_7,EIGHT	
	INC	ZERO_COUNTER	
	MOV	ZERO_BIT,#8	
EIGHT:	RET		

270068-19

Figure 18. Subroutine SCAN Determines Which of 8 Bits in LINE is Zero

Notice that RESPONSE_TO_KEY_CLOSURE does not change the Accumulator, the PSW, nor any of the registers R0 through R7. Neither do SCAN or DEBOUNCE_DELAY.

What we come out with then is a one-byte key address (ADDRESS) which identifies the pressed key. The key's scan line number is in the upper nibble of ADDRESS, and its receive line number is in the lower nibble. ADDRESS can be used in a look-up table to generate a key code to transmit to a host computer, and/or to a display device.

The keyboard interrupt itself must be edge-triggered, rather than level-activated, so that the interrupt routine is invoked when a key is pressed, and is not constantly being repeated as long as the key is held down. In edge-triggered mode, the on-chip hardware clears the interrupt flag (EX0, in this case) as the service routine is being vectored to. In this application, however, contact bounce will cause several more edges to occur after the service routine has been vectored to, during the DEBOUNCE_DELAY routine. Consequently it is necessary to clear EX0 again in software before executing RETI.

The debounce delay routine also takes advantage of the Idle mode. In this routine a timer must be preloaded with a value appropriate to the desired length of delay. This would be

For example, with a 3.58 MHz oscillator frequency, a 30 ms delay could be obtained using a preload value of -8950, or DD0A, in hex digits.

In the debounce delay routine (Figure 19), the timer interrupt is enabled and set to a higher priority than the keyboard interrupt, because as we invoke Idle, the keyboard interrupt is still "in progress". An interrupt of the same priority will not be acknowledged, and will not terminate the Idle mode. With the timer interrupt set to priority 1, while the keyboard interrupt is a priority 0, the timer interrupt, when it occurs, will be acknowledged and will wake up the CPU. The timer interrupt service routine does not itself have to do anything. The service routine might be nothing more than a single RETI instruction. RETI from the timer interrupt service routine then returns execution to the debounce delay routine, which shuts down the timer and returns execution to the keyboard service routine.

DRIVING AN LCD

An LCD (Liquid Crystal Display) consists of a backplane and any number of segments or dots which will be used to form the image being displayed. Applying a voltage (nominally 4 or 5V) between any segment and the backplane causes the segment to darken. The only catch is that the polarity of the applied voltage has to be periodically reversed, or else a chemical reac-

$$\text{timer preload} = \frac{(\text{osc kHz}) \times (\text{delay time ms})}{12}$$

```

DEBOUNCE_DELAY:
    MOV     TL1, #TL1_PRELOAD ; Preload low byte
    MOV     TH1, #TH1_PRELOAD ; Preload high byte
    SETB    ET1                 ; Enable Timer 1 interrupt.
    SETB    PT1                 ; Set Timer 1 interrupt to high priority.
    SETB    TR1                 ; Start timer running.
    ORL     PCON, #1           ; Invoke Idle mode.

    ; The next instruction will not be executed until the delay times out.

    CLR     TR1                 ; Stop the timer.
    CLR     PT1                 ; Back to priority 0 (if desired).
    CLR     ET1                 ; Disable Timer 1 interrupt (if desired).
    RET                     ; Continue keyboard scan.

```

270068-20

Figure 19. Subroutine DEBOUNCE_DELAY Puts the 80C51BH into Idle During the Delay Time

tion takes place in the LCD which causes deterioration and eventual failure of the liquid crystal.

To prevent this happening, the backplane and all the segments are driven with an AC signal, which is derived from a rectangular voltage waveform. If a segment is to be "off" it is driven by the same waveform as the backplane. Thus it is always at backplane potential. If the segment is to be "on" it is driven with a waveform that is the inverse of the backplane waveform. Thus it has about 5V of periodically changing polarity between it and the backplane.

With a little software overhead, the 80C51BH can perform this task without the need for additional LCD drivers. The only drawback is that each LCD segment uses up one port pin, and the backplane uses one more. If more than, say, two 7-segment digits are being driven, there aren't many port pins left for other tasks. Nevertheless, assuming a given application leaves enough port pins available to support this task, the considerations for driving the LCD are as follows.

Suppose, for example, it is a 2-digit display with a decimal point. One port (TENS_DIGIT) connects to the 7 segments of the tens digit plus the backplane. Another port (ONES_DIGIT) connects to a decimal point plus the 7 segments of the ones digit.

One of the 80C51BH's timers is used to mark off half-periods of the drive voltage waveform. The LCD drive waveform should have a rep rate between 30 and 100 Hz, but it's not very critical. A half-period of 12 ms will set the rep rate to about 42 Hz. The preload/reload value to get 12 ms to rollover is the 2's complement negative of the oscillator frequency in kHz: if the oscillator frequency is 3.58 MHz, the reload value is -3580, or F204 in hex digits.

Now, the 80C51BH would normally be in Idle, to conserve power, during the time that the LCD and other

tasks are not requiring servicing. When the timer rolls over it generates an interrupt, which brings the 80C51BH out of Idle. The service routine reloads the timer (for the next rollover), and inverts the logic levels of all the pins that are connected to the LCD. It might look like this:

```

LCD_DRIVE_INTERRUPT:
    MOV     TL1, #LOW( - XTAL_FREQ)
    MOV     TH1, #HIGH( - XTAL_FREQ)
    XRL     TENS_DIGIT, #OFFH
    XRL     ONES_DIGIT, #OFFH
    RETI

```

To update the display, one would use a look-up table to generate the characters. In the table, "on" segments are represented as 1s, and "off" segments as 0s. The backplane bit is represented as a 0. The quantity to be displayed is stored in RAM as a BCD value. The look-up table operates on the low nibble of the BCD value, and produces the bit pattern that is to be written to either the ones digit or the tens digit. Before the new patterns can be written to the LCD, the LCD drive interrupt has to be disabled. That is to prevent a polarity reversal from taking place between the times the two digits are written. An update subroutine is shown in Figure 20.

USING AN LCD DRIVER

As was noted, driving an LCD directly with an 80C51BH uses a lot of port pins. LCD drivers are available in CMOS to interface an 80C51BH to a 4-digit display using only 7 of the C51BH's I/O pins. Basically, the C51BH tells the LCD driver what digit is to be displayed (4 bits) and what position it is to be displayed in (2 bits), and toggles a Chip Select pin to tell the driver to latch this information. The LCD driver generates the display characters (hex digits), and takes care of the polarity reversals using its own RC oscillator to generate the timing.

Figure 20 shows an 80C51BH working with an ICM7211M to drive a 4-digit LCD, and the software that updates the display.

One could equally well send information to the LCD driver over the bus. In that case, one would set up the Accumulator with the digit select and data input bits, and execute a MOVX@ R0,A instruction. The LCD driver's chip select would be driven by the CPU's WR signal. This is a little easier in software than the direct bit manipulation shown in Figure 21. However, it uses more I/O pins, unless there is already some external memory involved. In that case, no extra pins are used up by adding the LCD driver to the bus.

RESONANT TRANSDUCERS

Analog transducers are often used to convert the value of a physical property, such as temperature, pressure, etc., to an analog voltage. These kinds of transducers then require an analog-to-digital converter to put the measurement into a form that is compatible with a digital control system. Another kind of transducer is now becoming available that encodes the value of the physical property into a signal that can be directly read by a digital control system. These devices are called resonant transducers.

Resonant transducers are oscillators whose frequency depends in a known way on the physical property being measured. These devices output a train of rectangular pulses whose repetition rate encodes the value of the quantity being measured. The pulses can in most cases be fed directly into the 80C51BH, which then measures either the frequency or period of the incoming signal, basing the measurement on the accuracy of its own clock oscillator. The 80C51BH can even do this in its sleep; that is, in Idle.

When the frequency or period measurement is completed, the C51BH wakes itself up for a very short time to perform a sanity check on the measurement and convert it in software to any scaling of the measured quantity that may be desired. The software conversion can include corrections for nonlinearities in the transducer's transfer function.

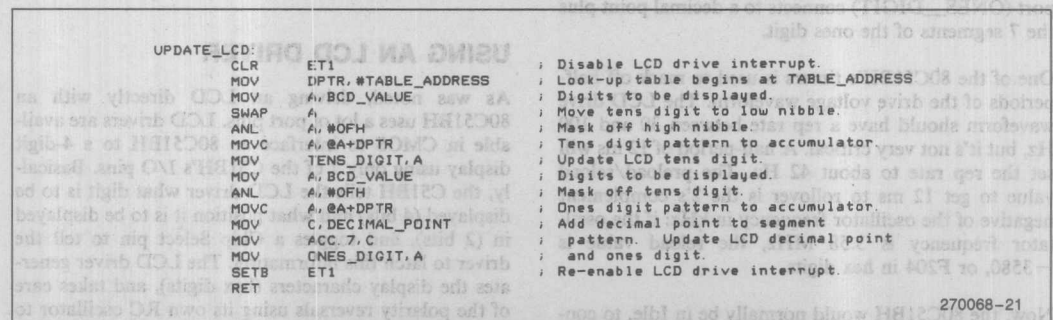
Resolution is also controlled by software, and can even be dynamically varied to meet changing needs as a situation becomes more critical. For example, in a process controller you can increase your resolution ("fine tune" the control, as it were) as the process approaches its target.

The nominal reference frequency of the output signal from these devices is in the range of 20 Hz to 500 kHz, depending on the design. Transducers are available that have a full scale frequency shift 2 to 1. The transducer operates from a supply voltage range of 3V to 20V, which means it can operate from the same supply voltage as the 80C51BH. At 5V, the transducer draws less than 5 mA (Reference 7). It can normally be connected directly to one of the C51BH's port pins, as shown in Figure 22.

FREQUENCY MEASUREMENTS

Measuring a frequency means counting pulses for a known sample time. Two timer/counters can be used, one to mark off the sample time and one to count pulses. If the frequency being counted doesn't exceed 50 kHz or so, one may equally well connect the transducer signal to one of the external interrupt pins, and count pulses in software. That frees up one timer, with very little cost in CPU time.

The count that is directly obtained is TxF, where T is the sample time and F is the frequency. The full scale



270068-21

Figure 20. UPDATE_LCD Routine Writes Two Digits to an LCD

range is $T_x(F_{max}-F_{min})$. For n-bit resolution

$$1 \text{ LSB} = \frac{T_x(F_{max}-F_{min})}{2^n}$$

Therefore the sample time required for n-bit resolution is

$$T = \frac{2^n}{F_{max}-F_{min}}$$

For example, 8-bit resolution in the measurement of a frequency that varies between 7 kHz and 9 kHz would require, according to this formula, a sample time of 128 ms. The maximum acceptable frequency count would be $128 \text{ ms} \times 9 \text{ kHz} = 1152$ counts. The minimum would be 896 counts. Subtracting 896 from each frequency count (or presetting the frequency counter to $-896 = 0FC80H$) would allow the frequency to be reported on a scale of 0 to FF in hex digits.

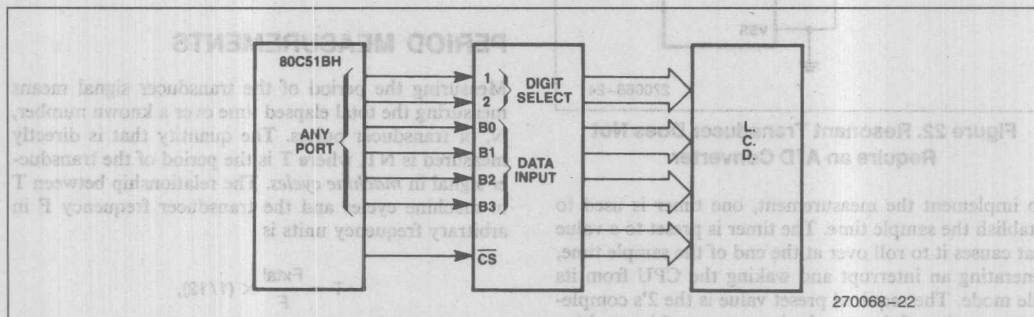


Figure 21a. Using an LCD Driver

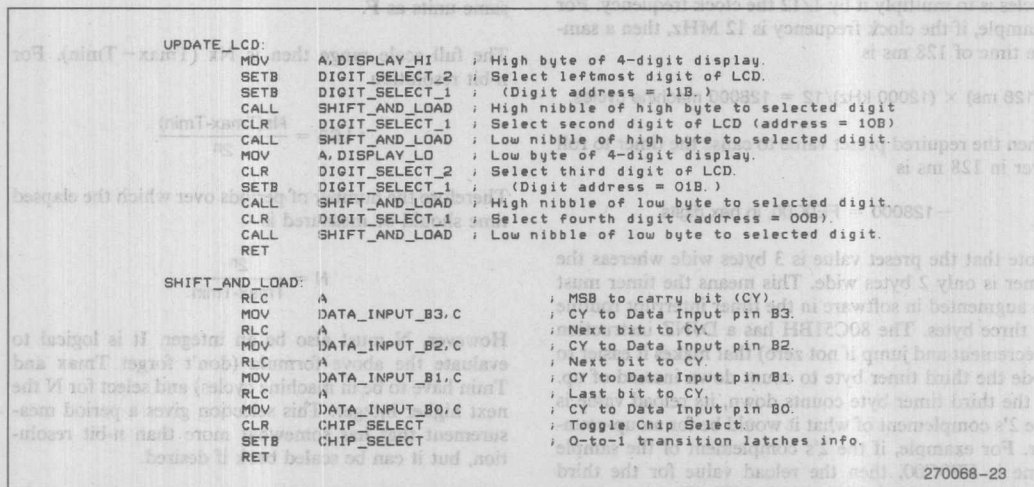


Figure 21b. UPDATE_LCD Routine Writes 4 Digits to an LCD Driver

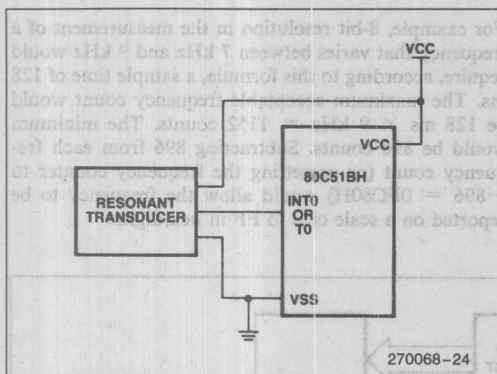


Figure 22. Resonant Transducer Does Not Require an A/D Converter

To implement the measurement, one timer is used to establish the sample time. The timer is preset to a value that causes it to roll over at the end of the sample time, generating an interrupt and waking the CPU from its Idle mode. The required preset value is the 2's complement negative of the sample time measured in machine cycles. The conversion from sample time to machine cycles is to multiply it by 1/12 the clock frequency. For example, if the clock frequency is 12 MHz, then a sample time of 128 ms is

$$(128 \text{ ms}) \times (12000 \text{ kHz})/12 = 128000 \text{ machine cycles.}$$

Then the required preset value to cause the timer to roll over in 128 ms is

$$-128000 = \text{FE0C00, in hex digits.}$$

Note that the preset value is 3 bytes wide whereas the timer is only 2 bytes wide. This means the timer must be augmented in software in the timer interrupt routine to three bytes. The 80C51BH has a DJNZ instruction (decrement and jump if not zero) that makes it easier to code the third timer byte to count down instead of up. If the third timer byte counts down, its reload value is the 2's complement of what it would be for an up-counter. For example, if the 2's complement of the sample time is FE0C00, then the reload value for the third timer byte would be 02, instead of FE. The timer interrupt routine might then be:

```
TIMER_INTERRUPT_ROUTINE:
    DNJZ    THIRD_TIMER_BYTE,OUT
    MOV     TLO,#0
    MOV     TH0,#0CH
    MOV     THIRD_TIMERBYTE,#2
    MOV     FREQUENCY,COUNTER_LO
;Preset COUNTER to -896:
    MOV     COUNTER_LO,#80H
    MOV     COUNTER_HI,#0FCH
OUT:      RETI
```

At this point the value of the frequency of the transducer signal, measured to 8 bit resolution, is contained in FREQUENCY. Note that the timer can be reloaded on the fly. Note too that the timer can be reloaded on the fly. Note too that for 8-bit resolution only the low byte of the frequency counter needs to be read, since the high byte is necessarily 0. However, one may want to test the high byte to ensure that it is zero, as a sanity check on the data. Both bytes, of course must be reloaded.

PERIOD MEASUREMENTS

Measuring the period of the transducer signal means measuring the total elapsed time over a known number, N, of transducer pulses. The quantity that is directly measured is NT, where T is the period of the transducer signal in machine cycles. The relationship between T in machine cycles and the transducer frequency F in arbitrary frequency units is

$$T = \frac{F_{\text{xtal}}}{F} \times (1/12),$$

where Fxtal is the 80C51BH clock frequency, in the same units as F.

The full scale range then is Nx (Tmax - Tmin). For n-bit resolution.

$$1 \text{ LSB} = \frac{N_s(T_{\text{max}} - T_{\text{min}})}{2^n}.$$

Therefore the number of periods over which the elapsed time should be measured is

$$N = \frac{2^n}{T_{\text{max}} - T_{\text{min}}}$$

However, N must also be an integer. It is logical to evaluate the above formula (don't forget Tmax and Tmin have to be in machine cycles) and select for N the next higher integer. This selection gives a period measurement that has somewhat more than n-bit resolution, but it can be scaled back if desired.

For example, suppose we want 8-bit resolution in the measurement of the period of a signal whose frequency varies from 7.1 kHz to 9 kHz. If the clock frequency is 12 MHz, then Tmax is (12000 kHz/7.1 kHz) x (1/12) = 141 machine cycles. Tmin is 111 machine cycles. The required value for N, then, is 256/(141-111) = 8.53 periods, according to the formula. Using N = 9 periods will give a maximum NT value of 141 x 9 = 1269 machine cycles. The minimum NT will be 111 x 9 = 999 machine cycles. A lookup table can be used to

scale these values back to a range of 0 to 255, giving precisely the 8-bit resolution desired.

To implement the measurement, one timer is used to measure the elapsed time, NT. The transducer is connected to one of the external interrupt pins, and this interrupt is configured to the transition-activated mode. In the transition-activated mode every 1-to-0 transition in the transducer output will generate an interrupt. The interrupt routine counts transducer pulses, and when it gets to the predetermined N, it reads and clears the timer. For the specific example cited above, the interrupt routine might be:

```

INTERRUPT_RESPONSE:
    DJNZ    N,OUT
    MOV     N,#9
    CLR     EA
    CLR     TR1
    MOV     NT_LO,TL1
    MOV     NT_HI,TH1
    MOV     TL1,#9
    MOV     TH1,#0
    SETB    TR1
    SETB    EA
    CALL    LOOKUP_TABLE
OUT:      RETI

```

In this routine a pulse counter N is decremented from its preset value, 9, to zero. When the counter gets to zero it is reloaded to 9. Then all interrupts are blocked for a short time while the timer is read and cleared. The timer is stopped during the read and clear operations, so "clearing" it actually means presetting it to 9, to make up for the 9 machine cycles that are missed while the timer is stopped.

The subroutine LOOKUP_TABLE is used to scale the measurement back to the desired 8-bit resolution. It can also include built-in corrections for errors or nonlinearities in the transducer's transfer function.

The subroutine uses the MOVC A, @ A + DPTR instruction to access the table, which contains 270 entries commencing at the 16-bit address referred to as TABLE. The subroutine must compute the address of the table entry that corresponds to the measured value of NT. This address is

$$DPTR = TABLE + NT - NTMIN,$$

where NTMIN = 999, in this specific example.

```

LOOKUP_TABLE:
    PUSH    ACC
    PUSH    PSW
    MOV     A,#LOW(TABLE-NTMIN)
    ADD     A,NT_LO
    MOV     DPL,A
    MOV     A,#HIGH(TABLE-NTMIN)

```

```

    ADDC    A,NT_HI
    MOV     DPH,A
    CLR     A
    MOVC    A,@A+DTPR
    MOV     PERIOD,A
    POP     PSW
    POP     ACC
    RET

```

At this point the value of the period of the transducer signal, measured to 8 bit resolution, is contained in PERIOD.

PULSE WIDTH MEASUREMENTS

The 80C51BH timers have an operating mode which is particularly suited to pulse width measurements, and will be useful in these applications if the transducer signal has a fixed duty cycle.

In this mode the timer is turned on by the on-chip circuitry in response to an input high at the external interrupt pin, and off by an input low, and it can do this while the 80C51BH is in Idle. (The "GATE" mode of timer operation is described in the Intel Microcontroller Handbook.) The external interrupt itself can be enabled, so the same 1-to-0 transition from the transducer that turns off the timer also generates an interrupt. The interrupt routine then reads and resets the timer.

The advantage of this method is that the transducer signal has direct access to the timer gate, with the result that variations in interrupt response time have no effect on the measurement.

Resonant transducers that are designed to fully exploit the GATE mode have an internal divide-by-N circuit that fixes the duty cycle at 50% and lowers the output frequency to the range of 250 to 500 Hz (to control RFI). The transfer function between transducer period and measurand is approximately linear, with known and repeatable error functions.

HMOS/CHMOS Interchangeability

The CHMOS version of the 8051 is architecturally identical with the HMOS version, but there are nevertheless some important differences between them which the designer should be aware of. In addition, some applications require interchangeability between HMOS and CHMOS parts. The differences that need to be considered are as follows:

External Clock Drive: To drive the HMOS 8051 with an external clock signal, one normally grounds the XTAL1 pin and drives the XTAL2 pin. To drive the CHMOS 8051 with an external clock signal, one must drive the XTAL1 pin and leave the XTAL2 pin unconnected. The reason for the difference is that in the

HMOS 8051, it is the XTAL2 pin that drives the internal clocking circuits, whereas in the CHMOS version it is the XTAL1 pin that drives the internal clocking circuits.

There are several ways to design an external clock drive to work with both types. For low clock frequencies (below 6 MHz), the HMOS 8051 can be driven in the same way as the CHMOS version, namely, through XTAL1 with XTAL2 unconnected. Another way is to drive both XTAL1 and XTAL2; that is, drive XTAL1 and use an external inverter to derive from XTAL1 a signal with which to drive XTAL2.

In either case, a 74HC or 74HCT circuit makes an excellent driver for XTAL1 and/or XTAL2, because neither the HMOS nor the CHMOS XTAL pins have TTL-like input logic levels.

Unused Pins: Unused pins of Ports 1, 2 and 3 can be ignored in both HMOS and CHMOS designs. The internal pullups will put them into a defined state. Unused Port 0 pins in 8051 applications can be ignored, even if they're floating. But in 80C51BH applications, these pins should not be left afloat. They can be externally pulled up or down, or they can be internally pulled down by writing 0s to them.

8031/80C31BH designs may or may not need pullups on Port 0. Pullups aren't needed for program fetches, because in bus operations the pins are actively pulled high or low by either the 8031 or the external program memory. But they are needed for the CHMOS part if the Idle or Power Down mode is invoked, because in these modes Port 0 floats.

Logic Levels: If V_{CC} is between 4.5V and 5.5V, an input signal that meets the HMOS 8051's input logic levels will also meet the CHMOS 80C51BH's input logic levels (except for XTAL1/XTAL2 and RST). For the same V_{CC} condition, the CHMOS device will reach or surpass the output logic levels of the HMOS device. The HMOS device will not necessarily reach the output logic levels of the CHMOS device. This is an important consideration if HMOS/CHMOS interchangeability must be maintained in an otherwise CMOS system.

HMOS 8051 outputs that have internal pullups (Ports 1, 2, and 3) "typically" reach 4V or more if I_{OH} is zero, but not fast enough to meet timing specs. Adding an external pullup resistor will ensure the logic level, but still not the timing, as shown in Figure 23. If timing is an issue, the best way to interface HMOS to CMOS is through a 74HCT circuit.

Idle and Power Down: The Idle and Power Down modes exist only on the CHMOS devices, but if one

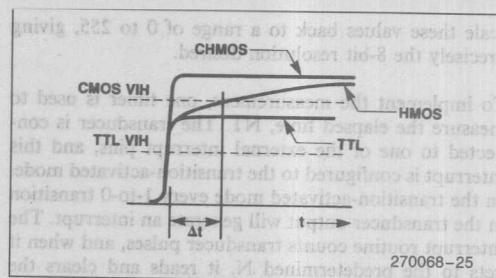


Figure 23. 0-to-1 Transition Shows Unspecified Delay (Δt) in HMOS to 74HC Logic

wishes to preserve the capability of interchanging HMOS and CHMOS 8051s the software has to be designed so that the HMOS parts will respond in an acceptable manner when a CHMOS reduced power mode is invoked.

For example, an instruction that invokes Power Down can be followed by a "JMP \$":

```
CLR    EA
ORL    PCON, #2
JMP    $
```

The CHMOS and HMOS parts will respond to this sequence of code differently. The CHMOS part, going into a normal CHMOS Power Down Mode, will stop fetching instructions until it gets a hardware reset. The HMOS part will go through the motions of executing the ORL instruction, and then fetch the JMP instruction. It will continue fetching and executing JMP \$ until hardware reset.

Maintaining HMOS/CHMOS 8051 interchangeability in response to Idle requires more planning. The HMOS part will not respond to the instruction that puts the CHMOS part into Idle, so that instruction needs to be followed by a software idle. This would be an idling loop which would be terminated by the same conditions that would terminate the CHMOS's hardware Idle. Then when the CHMOS device goes into Idle, the HMOS version executes the idling loop, until either a hardware reset or an enabled interrupt is received. Now if Idle is terminated by an interrupt, execution for the CHMOS device will proceed after RETI from the instruction following the one that invoked Idle. The instruction following the one that invoked Idle is the idling loop that was inserted for the HMOS device. At this point, both the HMOS and CHMOS devices must be able to fall through the loop to continue execution.

One way to achieve the desired effect is to define a "fake" Idle flag, and set it just before going into Idle. The instruction that invoked Idle is followed by a software idle:

```
SETB   IDLE
ORL     PCON,#1
JB      IDLE,$
```

Now the interrupt that terminates the CHMOS's Idle must also break the software idle. It does so by clearing the "Idle" bit:

```
...
CLR     IDLE
RETI
```

Note too that the PCON register in the HMOS 8051 contains only one bit, SMOD, whereas the PCON register in CHMOS contains SMOD plus four other bits. Two of those other bits are general purpose flags. Maintaining HMOS/CHMOS interchangeability requires that these flags not be used.

REFERENCES

1. Pawlowski, Moroyan, Alnether, "Inside CMOS Technology," *BYTE magazine*, Sept., 1983. Available as Article Reprint AR-302.
2. Kokkonen, Pashley, "Modular Approach to C-MOS Technology Tailors Process to Application," *Electronics*, May, 1984. Available as Article Reprint AR-332.
3. Williamson, T., *Designing Microcontroller Systems for Electrically Noisy Environments*, Intel Application Note AP-125, Feb. 1982.
4. Williamson, T., "PC Layout Techniques for Minimizing Noise," *Mini-Micro Southeast*, Session 9, Jan., 1984.
5. Alnether, J., *High Speed Memory System Design Using 2147H*, Intel Application Note AP-74, March 1980.
6. Ott, H., "Digital Circuit Grounding and Interconnection," *Proceedings of the IEEE Symposium on Electromagnetic Compatibility*, pp. 292-297, Aug. 1981.
7. *Digital Sensors by Technar*, Technar Inc., 205 North 2nd Ave., Arcadia, CA 91006.



APPLICATION NOTE

AP-410

- REFERENCES**
1. Pawlowski, Morton, "Inside CMOS Technology," BYTE magazine, Sept. 1983. Available as Article Reprint AR-303.
 2. Kokkonen, Tiesley, "Modular Approach to CMOS Technology," IBM Journal of Research and Development, May 1984. Available as Article Reprint AR-313.
 3. Williamson, T., "Designing Microcontroller Systems for Electrically Noisy Environments," Intel Application Note AN-113, Feb. 1983.
 4. Williamson, T., "PC Layout Techniques for Minimizing Noise," Min-Micro Systems, Section 9, Jan. 1984.
 5. Alachkar, J., "High Speed Memory Access Rates Using 3M3M Intel Application Note AN-74, March 1980.
 6. Ott, H., "Digital Circuit Grounding and Interconnection," Proceedings of the IEEE Symposium on Electromagnetic Compatibility, pp. 392-397, Aug. 1981.
 7. Digital Sensors by Techno-Tech, Inc., 305 North 2nd Ave., Arcadia, CA 91006.

One way to achieve the desired effect is to define a "fake" idle flag, and set it just before going into idle. The instruction that invoked idle is followed by a software idle:

```
SETB   IDLE
ORL     PCON, #1
IDLE:  $
```

Now the interrupt that terminates the CMOS's idle must also break the software idle. It does so by clearing the "idle" bit.

```
CLR     IDLE
```

November 1987

Note too that the PCON register in the HMOS 8051 contains only one bit, SMOD, whereas the PCON register in CHMOS contains SMOD plus four other bits. Two of those other bits are general purpose flags. Maintaining HMOS/CHMOS interchangeability requires that these flags not be used.

Enhanced Serial Port on the 83C51FA

BETSY JONES
ECO APPLICATIONS ENGINEER

Order Number: 270490-001

The serial port on the 8051 has been enhanced on the 83C51FA with the addition of two new features: Automatic Address Recognition and Framing Error Detection. Automatic Address Recognition facilitates multiprocessor communications by reducing CPU overhead. Framing Error Detection increases communication reliability by checking each reception for a valid stop bit.

This Application Note explains how to use these new features with samples of code for typical applications. A section is also included which reviews how to set up the serial port for multiprocessor applications.

MULTIPROCESSOR COMMUNICATIONS

In applications where multiple controllers jointly perform a task, the master controller must be able to communicate selectively with individual slaves. To do this, the master first identifies the target slave (or slaves) with an address byte and then transmits a block of data. The target slaves must be able to identify their own address before receiving any data bytes.

The serial port on the 8051 provides a 9-bit mode to facilitate multiprocessor communication. The 9th bit allows the controller to distinguish between address and data bytes. In this mode, a total of 11 bits are received or transmitted: a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). See Figure below.

The 9th bit is set to 1 to identify address bytes and set to 0 for data bytes. A typical data stream is seen below:

ADDRESS BYTE / DATA BYTE / DATA BYTE / ...
D8 = 1 D8 = 0 D8 = 0

Initially the slave is set up to only receive address bytes. Once it receives its own address, the slave reconfigures itself to receive data. On the 8051 serial port, an address byte interrupts all slaves for an address comparison. On the 83C51FA, however, Automatic Address Recognition allows the addressed slave to be the only one interrupted; that is, the address comparison occurs in hardware, not software. With this feature, the master controller can establish communication with one or more slaves without all the slaves having to respond to the transmission.

AUTOMATIC ADDRESS RECOGNITION

Automatic Address Recognition reduces the CPU time required to service the serial port. Since the CPU is only interrupted when it receives its own address, the software overhead to compare addresses is eliminated. This would also effectively reduce the sophistication of the serial protocol when numerous controllers are sharing the same serial link.

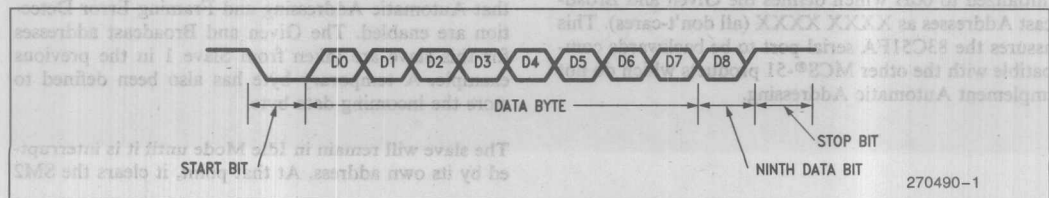
This same feature can also be used in conjunction with the Idle Mode to reduce the system's overall power consumption. For instance, a master may need to communicate with only one slave at a time. With all slaves in Idle Mode, only that one slave would be interrupted to respond to the master's transmission. Without Automatic Addressing, each slave would have to "wake up" to check for its address. Limiting the interruptions reduces the amount of current drawn by the system and thus reduces the power consumption.

In multiprocessor applications the serial port is configured in either of the 9-bit modes (Mode 2 or 3). Mode 2 has a fixed baud rate whereas Mode 3 is variable. For more information on the different serial port modes refer to the "Serial Port Set Up" section.

Automatic Address Recognition is enabled by setting the SM2 bit in SCON. Each slave has its SM2 bit set waiting for an address byte (9th bit = 1). The Receive Interrupt (RI) flag will get set when the received byte corresponds to either a Given or Broadcast Address. The slave then clears its SM2 bit to enable reception of data bytes (9th bit = 0) from the master.

The master can selectively communicate with groups of slaves by using the Given Address. Addressing all slaves at once is possible with the Broadcast Address. These addresses are defined for each slave by two new Special Function Registers: SADDR and SADEN.

A slave's individual address is specified in SADDR. SADEN is a mask byte that defines don't-cares to form the Given Address. These don't-cares allow flexibility in the user-defined protocol to address one or more slaves. The following is an example of how to define Given Addresses and selectively address different slaves.



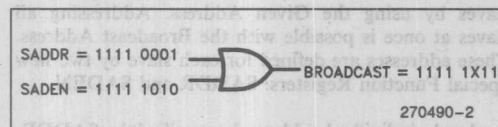
Slave 1			
SADDR	=	1111	0001
SADEN	=	1111	1010
GIVEN	=	1111	0X0X
Slave 2			
SADDR	=	1111	0011
SADEN	=	1111	1001
GIVEN	=	1111	0XX1

The SADEN bytes have been selected such that bit 1 (LSB) is a don't-care for Slave 1's Given Address, but bit 1 = 1 for Slave 2. Thus, to selectively communicate with just Slave 1 an address with bit 1 = 0 would be used (e.g. 1111 0000).

Similarly, bit 2 = 0 for Slave 1, but is a don't-care for Slave 2. Now to communicate with just Slave 2 an address with bit 2 = 1 would be used (e.g. 1111 0111).

Finally, to communicate with both slaves at once the address must have bit 1 = 1 and bit 2 = 0. Notice, however, that bit 3 is a "don't-care" for both slaves. This allows two different addresses to select both slaves (1111 0001 or 1111 0101). If a third slave was added that required its bit 3 = 0, then the latter address could be used to communicate with Slave 1 and 2 but not Slave 3.

The master can also communicate with all slaves at once with the Broadcast Address. It is formed from the logical OR of SADDR and SADEN with zeros defined as don't-cares. For example, the Broadcast address for Slave 1 would be formed as follows:



The don't-cares also allow flexibility in defining the Broadcast Address, but in most applications a Broadcast Address will be 0FFH.

SADDR and SADEN are located at address A9H and B9H, respectively. On Reset, SADDR and SADEN are initialized to 00H which defines the Given and Broadcast Addresses as XXXX XXXX (all don't-cares). This assures the 83C51FA serial port to be backwards compatible with the other MCS®-51 products which do not implement Automatic Addressing.

FRAMING ERROR DETECTION

Framing Error Detection is another new feature on 83C51FA serial port which allows the receiving controller to check for valid stop bits in Modes 1, 2, or 3. A missing stop bit can be caused, for example, by noise on the serial lines or transmission by two CPUs simultaneously.

If a stop bit is missing a Framing Error bit FE will be set. This bit can then be checked in software after each reception to detect communication errors. Once set, the FE bit must be cleared in software. A valid stop bit will not clear FE.

The FE bit is located in SCON and shares the same bit address as SM0. To determine which is accessed, a new control bit called SMOD0 has been added in the PCON register (see figures below). If SMOD0 = 0, then accesses to SCON.7 are to SM0. If SMOD0 = 1, then accesses to SCON.7 are to FE.

PCON: Power Control Register (Not Bit Addressable)

SMOD1	SMOD0	—	POF	GF1	GF0	PD	IDL
-------	-------	---	-----	-----	-----	----	-----

Address = 87H

SCON: Serial Port Control Register (Bit Addressable)

SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI
--------	-----	-----	-----	-----	-----	----	----

Address = 98H

SERIAL PORT SOFTWARE

The following sections of code show examples of how to invoke Automatic Addressing and Framing Error Detection. Routines for both the slave and master are given. Code is also included to initialize both serial ports; however, for more information on setting up the serial port refer to the next section.

For this example, the master and slave are transmitting/receiving at 9600 baud with a 12 MHz crystal frequency. To obtain this baud rate, the serial port is configured in Mode 3 and Timer 2 is used as the baud-rate generator.

Listing 1 shows the initialization for the slave. Notice that Automatic Addressing and Framing Error Detection are enabled. The Given and Broadcast addresses for this slave are taken from Slave 1 in the previous example. A temporary byte has also been defined to store the incoming data byte.

The slave will remain in Idle Mode until it is interrupted by its own address. At that point, it clears the SM2

Listing 1. Initialization Routine for the Slave

```

ORG    00H
LJMP   INIT

ORG    0023H
LJMP   SERIAL_PORT_INTERRUPT

TEMP   DATA   30H      ; Temporary storage byte

INIT:  MOV   SCON, #0FH      ; Mode 3, enable Auto Addressing
                                ; and reception
        ORL   PCON, #40H      ; FE bit accessed (SMOD0 = 1)
        MOV   RCAP2H, #0FFH    ; Reload values for 9600 Baud
        MOV   RCAP2L, #0D9H
        MOV   T2CON, #34H      ; Timer 2 set up, TR2 = 1 turns
                                ; timer on

INTERRUPTS:  SETB EA      ; Enable global interrupt
              SETB ES      ; Enable serial port interrupt

ADDRESSES:  MOV   SADDR, # 11110001 ; Define Given & Broadcast
              MOV   SADEN, # 11111010 ; Addresses
                                ; GIVEN    = 11110X0X
                                ; BROADCAST = 11111X11

IDLE_MODE:  ORL   PCON, #01H      ; Invoke Idle Mode

```

Listing 2. Receive Routine for the Slave

```

SERIAL_PORT_INTERRUPT:
    PUSH PSW
    CLR RI      ; RI set when address is
                ; recognized & must be cleared
                ; in software
                ; Reconfigure slave to receive
                ; data bytes

    CLR SM2      ; Wait for RI to be set
                ; Check for framing error

    RECEIVE_DATA:
        JNB RI, $      ; Wait for RI to be set
        MOV C, SCON.7    ; Check for framing error
        JC FRAMING_ERROR
        MOV TEMP, SBUF    ; Receive data byte & store
                        ; in temporary location
        CLR RI      ; Clear flag for next
                        ; reception
        SETB SM2      ; Re-enable Automatic
                        ; Addressing

        POP PSW
        RETI

    FRAMING_ERROR:
        CLR SCON.7      ; Clear FE bit
        CLR C
        .
        .
        .
        POP PSW
        RETI

```

Listing 3. Initialization and Transmit Routines for the Master

```

GIVEN_1      equ      11110001B
MESSAGE_1    data     30H

INIT:        MOV SCON, #0DOH      ; Mode 3, REN = 1
             MOV RCAP2H, #OFFH    ; 9600 Baud
             MOV RCAP2L, #0D9H
             MOV T2CON, #34H      ; Timer 2 set up, TR2 = 1

TRANSMIT_ADDRESS:
CLR TI
SETB TB8      ; Mark 1st byte as an address
              ; byte (9th bit = 1)
MOV SBUF, #GIVEN_1 ; Send address
JNB TI, $      ; Wait for transmission
              ; complete
CLR TI        ; Clear flag for next
              ; transmission

TRANSMIT_DATA:
CLR TB8      ; Mark 2nd byte as a data
              ; byte (9th bit = 0)
MOV SBUF, MESSAGE_1 ; Send data byte
JNB TI, $
CLR TI

```

bit to enable reception of data bytes. Depending on the user's protocol, more than one data byte may actually be received. This example, however, assumes only one byte of data follows each address byte.

Listing 2 shows the receive routine. Notice that when the data byte is received, the software checks for a framing error. The error routine could, for example, send an error message to the master and ask the master to re-transmit the last message. Before exiting the routine the SM2 is set to 1 to reenable Automatic Addressing. Once the slave has responded to the master's command, it could also put itself back into Idle Mode to wait for the next message.

The initialization routine for the master in Listing 3 is very similar to the slave. In this example, however, the master does not need Automatic Addressing; it is simply transmitting address and data bytes. GIVEN_1 is a byte to address the slave in the above example. MESSAGE_1 is a register that contains the data byte sent to this slave. Its value is arbitrary for the sample code.

SERIAL PORT SET UP

This section describes how to initialize the 83C51FA serial port for multiprocessor applications. Two different modes are available which provide 9-bit operation:

Mode 2 which has a fixed baud rate and Mode 3 which has a variable baud rate. Baud rates can be generated by either Timer 1 or Timer 2 (available on the 83C51FA but not the 8051). Deciding which mode and timer to use is determined by the desired baud rate and clock frequency of the particular application.

Another consideration is the tolerance needed between serial ports. Since the serial port re-synchs its receiver at every start bit, only 8 or 9 bit-times are available to accumulate timing errors. As a result, the receiver and transmitter only have to be within about 5% of each other's baud rate. Allowing equal error to both transmitter and receiver, only about 2% accuracy is actually needed.

Following is a discussion of both Modes 2 and 3 and examples of how to program each. The mode selection bits (SM0 and SM1) are located in SCON. The REN bit must also be set to enable reception.

SCON: Serial Port Control Register (Bit Addressable)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
-----	-----	-----	-----	-----	-----	----	----

Address = 98H

Mode	SM0	SM1	Baud Rate
2	1	0	Fosc/64 or Fosc/32
3	1	1	Variable

Example 1. Serial Port Mode 2

; Frequency = 12 MHz	
; Desired Baud Rate = 375 kBaud	
; = 1/32 (Osc Freq)	
MOV SCON, #0B0H	; Serial port Mode 2
	; Automatic Addressing (SM2 = 1),
	; reception enabled (REN = 1)
ORL PCON, #80H	; SMOD1 = 1 to double baud rate

Mode 2

Mode 2 uses a fixed baud rate of 1/32 or 1/64 of the oscillator frequency depending on the value of the SMOD1 bit in PCON. This mode basically offers a choice of two high-speed baud rates. With a 12 MHz clock frequency, baud rates of 187.5 kbaud or 375 kbaud can be obtained.

None of the timer/counters need to be set up for Mode 2. Only the SFRs SCON and PCON need to be defined.

PCON: Power Control Register (Not Bit Addressable)

SMOD1	SMOD0	—	POF	GF1	GF0	PD	IDL
-------	-------	---	-----	-----	-----	----	-----

Address = 87H

The baud rate in this mode is calculated by:

$$\text{Mode 2 Baud Rate} = \frac{2\text{SMOD1} \times \text{Osc Freq}}{64}$$

SMOD1 = 0, Baud Rate = 1/64 Osc Freq

SMOD1 = 1, Baud Rate = 1/32 Osc Freq

Mode 3

Mode 3 of the serial port has a variable baud rate generated by either Timer 1 or Timer 2. The baud rate is generated by the rollover rate of the selected timer. The timer is operated in an auto-reload mode so it will roll over to the reload value selected in software.

Baud rates based off Timer 2 have less granularity so that almost any baud rate can be obtained at a given clock frequency. However, Timer 1 is sufficient if the desired baud rate can be obtained at the specified clock frequency. Remember baud rates only need about 2% accuracy.

Timer 1 Set Up

To generate baud rates Timer 1 is usually configured in 8-bit auto-reload mode (Mode 2). The mode select bits

are M1 and M0 located in TMOD. To turn on Timer 1 the TR1 bit in TCON must be set. Also, the Timer 1 interrupt should be disabled in this application so that when the timer overflows it does not generate an interrupt.

TMOD: Timer/Counter Mode Control Register
(Not bit addressable)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
------	-----	----	----	------	-----	----	----

Timer 1

Timer 0

Address = 89H

TCON: Timer/Counter Control Register
(Bit addressable)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
-----	-----	-----	-----	-----	-----	-----	-----

Address = 88H

The formula for calculating the baud rate is given below. TH1 is the reload value for Timer 1 when it overflows.

$$\text{Baud Rate} = \frac{K \times \text{Osc Freq}}{32 \times 12 \times [256 - (\text{TH1})]}$$

K = 1 if SMOD1 = 0.

K = 2 if SMOD1 = 1. (SMOD1 is at PCON.7)

If the baud rate is known, the reload value TH1 can be calculated by:

$$\text{TH1} = 256 - \frac{K \times \text{Osc Freq}}{384 \times \text{Baud Rate}}$$

TH1 must be an integer value. Rounding off TH1 to the nearest integer may not produce the desired baud rate with the 2% accuracy required. In this case, another crystal frequency may have to be chosen.

Refer to Table 1 for timer reload values for commonly used baud rates.

Table 1. Commonly Used Baud Rates Generated by Timer 1

Baud Rate	Osc Freq	SMOD1	Timer 1	
			TMOD	Reload Value
62.5K	12 MHz	1	20	FFH
19.2K	11.06 MHz	1	20	FDH
9.6K	11.06 MHz	0	20	FDH
4.8K	11.06 MHz	0	20	FAH
2.4K	11.06 MHz	0	20	F4H
1.2K	11.06 MHz	0	20	E8H
300	6 MHz	0	20	CCH
110	6 MHz	0	20	72H

Example 2. Serial Port Mode 3, with Timer 1 as Baud-Rate Generator

; Frequency = 11.0 MHz	
; Desired Baud Rate = 19.2 kBaud	
; TH1 = 256 - $\frac{(2) \times (11.0 \times 10^6)}{(32) \times (12) \times (19200)}$	
; = 253 = FDH	
MOV SCON, #0F0H	; Serial port Mode 3, SM2 = 1,
ORL PCON, #80H	; REN = 1
MOV TMOD, #20H	; SMOD1 = 1
MOV TH1, #0FDH	; Timer 1 Mode 2
; Reload value for desired baud	
; rate	
SETB TR1	
; Turn on Timer 1	

It can be seen that the exact frequency to generate the standard baud rates (19.2K, 9600, 4800, etc.) is 11.06 MHz. However, it is not necessary to use this exact frequency. With a 2% tolerance any crystal value from 10.8 MHz to 11.3 MHz is sufficient.

Timer 2 Set Up

Timer 2 has a special baud-rate generator mode which transmits and receives at the same baud rate. This mode is invoked by setting both the RCLK and TCLK bits in T2CON. To turn Timer 2 on the TR2 bit should also be set.

Unlike Timer 1, this mode does not require that the timer overflow interrupt be disabled. That is, when Timer 2 is in the baud-rate generator mode, its interrupt is disconnected from the Timer 2 overflow. This

interrupt then becomes available as a third external interrupt. (For more information on external interrupts, refer to the chapter "Hardware Description of the 8051" in the Embedded Controller Handbook.)

T2CON: Timer/Counter 2 Control Register
(Bit Addressable)

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
-----	------	------	------	-------	-----	------	--------

Address = C8H

This formula for calculating the baud rate is given below. (RCAP2H, RCAP2L) is the 16-bit reload value when Timer 2 overflows.

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{32 \times [65536 - (\text{RCAP2H}, \text{RCAP2L})]}$$

where (RCAP2H, RCAP2L) is a 16-bit unsigned integer.

To obtain the reload value for RCAP2H and RCAP2L the above equation can be rewritten as:

$$(RCAP2H, RCAP2L) = 65536 - \frac{\text{Osc Freq}}{32 \times \text{Baud Rate}}$$

Refer to Table 2 for reload values for commonly used baud rates.

Notice that when using Timer 2, most standard baud rates can be obtained at 12 MHz.

Table 2. Commonly Used Baud Rates Generated by Timer 2

Baud Rate	Osc Freq	Timer 2	
		RCAP2H	RCAP2L
375K	12 MHz	FF	FF
9.6K	12 MHz	FF	D9
4.8K	12 MHz	FF	B2
2.4K	12 MHz	FF	64
1.2K	12 MHz	FE	C8
300	12 MHz	FB	1E
110	12 MHz	F2	AF
300	6 MHz	FD	8F
110	6 MHz	F9	57

Example 3. Serial Port Timer with Timer 2 as Baud-Rate Generator

```

; Frequency      = 12 MHz
; Desired Baud Rate = 9600 Baud
;
;
; (RCAP2H, RCAP2L) = 65536 - ((12 x 106) / (32 x (9600)))
;                  = 65497 = FFD9H
;
MOV SCON, #0F0H ; Serial port Mode 3, SM2 = 1,
; REN = 1
MOV RCAP2H, #0FFH ; Reload values for desired
MOV RCAP2L, #0D9H ; baud rate
MOV T2CON, #34H ; Timer 2 as baud rate
; generator, turn on Timer 2

```

Using the 8051 Microcontroller with Resonant Transducers

TOM WILLIAMSON

Abstract—Having to interface an analog transducer to a digital control system through an analog-to-digital converter represents an expensive bottleneck in the development of many systems. Some transducer companies are addressing this problem by developing proprietary families of resonant transducers.

Resonant transducers are oscillators whose frequency depends in some known way on the physical property being measured. The electrical output from these devices is a train of rectangular pulses whose repetition rate encodes the value of the measurand. Changes in the measurand cause the frequency to shift. The microcontroller detects the frequency shift, runs a validity check on it, and converts it in software to the measurand value.

This paper discusses software interfacing techniques between resonant transducers and the 8051. Techniques for measuring frequency and period are discussed and compared for resolution and interrogation time. The 8051 is capable of performing these tasks in extremely short CPU time. Requirements for obtaining n -bit resolution in the measurement are discussed. It is determined that it is always faster to evaluate the measurand to a given level of resolution by measuring the period rather than the frequency, even if the measurand is proportional to the frequency rather than to the period. Numerical and software examples are presented to illustrate the concepts.

I. RESONANT TRANSDUCERS

MOST sensing transducers are not directly compatible with digital controllers, because they generate analog signals. A few transducer companies are developing proprietary families of sensors which generate signals that are more directly compatible with digital systems. These are not analog sensors with built-in A-D conversion, but oscillators whose frequency depends in some known way on the physical property being measured.

The technology is applicable to virtually any type of measurand: pressure, gas density, position, temperature, force, etc. The sensor and microcontroller can operate from the same supply voltage, so the sensor can in most cases connect directly to a port pin on the microcontroller.

The nominal reference frequency of the output signal from these devices is in the range of 20 Hz–500 kHz, depending on the design. A change in the measurand away from the reference condition causes the frequency to shift by an amount that is related to the change in the measurand value. Transducers are available that have a full-scale frequency shift of 2–1. The microcontroller detects the change in frequency or period and converts it in software to the measurand value.

II. CONNECTING THE DIGITAL TRANSDUCER TO THE 8051

Normally the transducer output can be connected directly to one of the 8051 port pins. An exception would occur when the

transducer signal does not restrict itself to the voltage range of -0.5 to $+5.5$ V.

The 8051 is not sensitive to the rise and fall times of its input signals. It detects transitions by sampling its port pins at fixed intervals (once per machine cycle), and responds to a change in the sequence of samples. If the slew rate of the transducer signal is extremely slow, noise superimposed on the signal could cause the sequence of samples to show false transitions. There could on that account be situations in which the transducer signal should be buffered through a Schmitt Trigger to square it up.

III. TIMER/COUNTER STRUCTURE IN THE 8051

The 8051 has two 16-bit timer/counters: Timer 0 and Timer 1. Both can be configured in software to operate either as timers or as event counters.

In the "timer" function, the register is automatically incremented every machine cycle. Since a machine cycle in the 8051 consists of 12 clock periods, the timer is being incremented at a constant rate of $1/12$ the clock frequency.

In the "counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin (70 or 71). The way this function works is the external input pin is sampled once each machine cycle (once every 12 clock periods), and when the samples show a high in one cycle and a low in the next, the count is incremented.

Note too that since it takes two machine cycles (24 clock periods) to recognize a 1-to-0 transition, the maximum count rate is $1/24$ the clock frequency. If the clock frequency is 12 MHz, the maximum count rate is 500 kHz. There are no requirements on the duty cycle of the signal being counted.

The 8052, an enhanced version of the 8051, has three 16-bit timer/counters, two of which are identical to those in the 8051. The third timer/counter can operate either as a 16-bit timer/counter with automatic reload to a preset 16-bit value on rollover, or as a 16-bit timer/counter with a "capture" mode. In the capture mode a 1-to-0 transition at the T2EX pin causes the current value in the counting register to the "captured" into RAM. The third timer makes the 8052 particularly easy to interface with resonant transducers.

IV. WHETHER TO MEASURE FREQUENCY OR PERIOD

Measuring the frequency requires counting transducer pulses for a fixed sample time. Measuring the period requires measuring elapsed time for a fixed number of transducer pulses. For a given level of accuracy in the determination of the value of the measurand, it is usually faster to measure the period, rather than the frequency, even if the measurand is

WILLIAMSON: USING THE 8051 MICROCONTROLLER

proportional to frequency rather than period. However, both types of measurements will be discussed here.

Two timer/counters can be used, one to mark time and the other to count transducer pulses. If the frequency being counted does not exceed 50 kHz or so, one may equally well connect the transducer signal to an external interrupt pin, and count transducer pulses in software. That frees one timer, with very little cost in CPU time.

V. HOW TO MEASURE TRANSDUCER FREQUENCY

Measuring the frequency means counting transducer pulses for some desired sample time. The count that is directly obtained is $T \times F$, where T is the sample time and F is the frequency. The full scale range is $T \times (F_{\max} - F_{\min})$. For n -bit resolution

$$1 \text{ LSB} = \frac{T \times (F_{\max} - F_{\min})}{2^n}$$

Therefore, the sample time required for n -bit resolution is

$$T = \frac{2^n}{F_{\max} - F_{\min}}$$

For example, 8-bit resolution in the measurement of a frequency that varies between 5 and 10 kHz would require, according to this formula, a sample time of 51.2 ms. The maximum acceptable frequency count would be 51.2 ms \times 10 kHz = 512 counts. The minimum would be 256 counts. Subtracting 256 from each frequency count would allow the frequency to be reported on a scale of 0 to FF in hex digits.

If F_{\min} and F_{\max} are closer together it takes more time to resolve them. 8-bit resolution in the measurement of a frequency that varies between 7 and 9 kHz would require a sample time of 128 ms. The maximum and minimum acceptable counts would be 1152 and 896. Subtracting 896 from each frequency count would allow the frequency to be reported on a scale of 0 to FF in hex digits.

To implement the measurement, one timer is used to establish the sample time. In this function it autoincrements every machine cycle. A machine cycle consists of 12 periods of the clock oscillator. The sample time can be converted to machine cycles by multiplying it by $(F_{\text{xtal}})/12$, where F_{xtal} is the 8051 clock frequency. The timer needs to be preset so that it rolls over at the end of each sample time. Then it generates an interrupt, and the interrupt routine reads and clears the transducer pulse counter, and then reloads the timer with the correct preset value.

The preset or reload value is the two's complement negative of the sample time in machine cycles. For example, with a 12-MHz clock frequency, the reload value required to establish a 51.2 ms sample time is

$$= \frac{(51.2 \text{ ms}) \times (12000 \text{ kHz})}{12} = -51200 = 3800 \text{ H.}$$

In many cases the required sample time exceeds the capacity of a 16-bit timer. For example, establishing a 128 ms sample time with a 12-MHz clock frequency requires a 3-byte timer with a reload of FE0C00H. The 8051 timer, being only 2-

bytes wide, can be augmented in software in the timer interrupt routine to three bytes. The 8051 has a DJNZ instruction (decrement and jump if not zero) which makes it easier to code the third timer byte to count down instead of up. If the third timer byte counts down, its reload value is the two's complement of what it would be for an up-counter. For example, if the two's complement of the sample time is FE0C00H, then the reload value for the third timer byte would be 02, instead of FE. The timer interrupt routine might then be

```
DJNZ THIRD_TIMER_BYTE,OUT
MOV  TL0,#0
MOV  TH0,#0CH
MOV  THIRD_TIMER_BYTE,#02
```

(Now read and clear the transducer pulse counter.)

OUT: RETI

Interrupt latency will have no effect on the measurement if the latency is the same for every sample time.

The trouble with measuring the frequency is it is not only slow, but a waste of the resolving power of the 8051's timers. A timer with microsecond resolution is being used to mark off 100-ms time periods. The technique is nevertheless useful if the timer is already serving other purposes (servicing a display, perhaps), so that the sample time is coming relatively free of charge. But in most cases it is faster and equally accurate to measure the frequency by deriving it from a measurement of the period.

VI. HOW TO MEASURE THE PERIOD

Measuring the period of the transducer signal means measuring the total elapsed time over N -transducer pulses. The quantity that is directly measured is $N \times T$, where T is the period of the transducer signal in machine cycles. The relationship between T in machine cycles and the transducer frequency F in arbitrary frequency units is

$$T = \frac{F_{\text{xtal}}}{F} \times (1/12)$$

where F_{xtal} is the 8051 clock frequency, in the same unit as F .

The full scale range then is $N \times (T_{\max} - T_{\min})$. For n -bit resolution

$$1 \text{ LSB} = \frac{N \times (T_{\max} - T_{\min})}{2^n}$$

Therefore, the number of periods over which the elapsed time should be measured is

$$N = \frac{2^n}{T_{\max} - T_{\min}}$$

However, N must also be an integer. It is logical to evaluate the above formula (do not forget that T_{\max} and T_{\min} have to be in machine cycles) and select for N the next higher integer. This selection gives a period measurement that has somewhat more than n -bit resolution, which may or may not be acceptable, depending on the overall requirements of the

system. It can be scaled back to n -bit resolution, if necessary, by the following computation:

$$\text{reported value} = \frac{NT - NT_{\min}}{NT_{\max} - NT_{\min}}$$

where NT is the elapsed time measured over N periods.

The computation can be done in math if a suitable divide routine is available in the software. For 8-bit resolution it is entirely reasonable to find the reported value in a look-up table, which would take up somewhat more than one page in ROM. In fact, the look-up table would contain $NT_{\max} - NT_{\min}$ entries.

For example, suppose we want 8-bit resolution in the measurement of the period of a signal whose frequency varies from 5 to 10 kHz. If the clock frequency is 12 MHz, then T_{\max} is $(12\,000\text{ kHz})/(12 \times 5\text{ kHz}) = 200$ machine cycles, and T_{\min} is 100 machine cycles. The timer needs to be on then for $N = 2.56$ periods, according to the formula. Using $N = 3$ periods will give maximum and minimum NT values of 600 and 300 machine cycles. This is somewhat more than 8-bit resolution. It can be scaled to 8 bits with a 300-byte look-up table, if desired.

To implement the measurement, one timer is used to measure the elapsed time NT . Enabling its interrupt is optional. The timer interrupt could be used to indicate a short or open in the transducer circuit.

Then the transducer is connected to one of the external interrupt pins (INT0 or INT1), and this interrupt is configured to the transition-activated mode. In the transition-activated mode every 1-to-0 transition in the transducer output will generate an interrupt. The interrupt routine counts transducer pulses, and when it gets to the predetermined N , it reads and clears the timer. For example

```
DJNZ PULSES,OUT
MOV PULSES,N,PERIODS
(Read and clear timer.)
```

```
OUT: RETI
```

If other interrupts are also to be enabled, the one connected to the transducer should be set to Priority 1, and the others to Priority 0. This is to control the interrupt response time. The response time will not affect the measurement if it is the same for every measurement. Variations in the response time will, however, affect the measurement. Setting the pulse-counter interrupt to Priority 1 and all others to Priority 0 will minimize variations in the response time. The response time will then be limited to range from 3 to 8 machine cycles.

VII. PULSEWIDTH MEASUREMENTS

The 8051 timers have an operating mode which is particularly suited to pulsewidth measurements, and may be useful here if the transducer has a fixed duty cycle, or if the transducer output is pulsewidth modulated instead of frequency modulated by the measurand.

In this mode the timer is turned on by the on-chip circuitry in response to an input high at the external interrupt pin, and off by an input low. The external interrupt itself is enabled, so the same 1-to-0 transition from the transducer that turns off the

timer also generates an interrupt. The interrupt routine would then read and reset the timer.

The advantage of this method is that the transducer signal has direct access to the timer gate, with the result that variations in the interrupt response time cease to be a factor. The timer can be read and cleared any time before the next high in the transducer output.

VIII. DERIVING FREQUENCY FROM A PERIOD MEASUREMENT

We now consider the problem of measuring the transducer frequency to n -bit resolution by deriving it from a direct measurement of the period. The advantage of this technique is speed. It is always faster to measure period than frequency. But it is important to end up with a frequency value that has the same resolution and accuracy as a directly measured frequency. Two questions need to be addressed.

- 1) To achieve n -bit resolution in the calculated frequency, how much resolution is required in the period?
- 2) Having measured the period to the required resolution, what is the most efficient way to calculate the frequency?

These questions will be addressed one at a time.

IX. RESOLUTION REQUIREMENTS

In general, n -bit resolution in the frequency derivation requires somewhat more than n -bit resolution in the period measurement. How much more? It will be demonstrated presently that if the transducer frequency varies over a 2-to-1 range, the frequency can be calculated with n -bit resolution from period measurements that have $(n + 1)$ -bit resolution.

The more practical form of the question is over how many periods (N) must the transducer signal be sampled to obtain the required resolution in F ? And so, we commence a calculation of N .

The basic calculation of frequency from $N \times T$ (which we shall call NT) is straightforward:

$$F = N/(NT)$$

The relationship between an increment dF in the calculated frequency due to an increment $d(NT)$ in the measured period is, therefore,

$$\begin{aligned} dF &= -\frac{N}{(NT)^2} d(NT) \\ &= -\frac{F^2}{N} d(NT). \end{aligned}$$

This equation says the value of an LSB in the calculated frequency is $(F^2)/N \times$ the value of an LSB in NT . Then the maximum value that an LSB in the calculated frequency can have is $(F_{\max})^2/N \times$ the value of an LSB in NT . For the calculated frequency to have n -bit resolution over the entire range of frequencies, the value of its LSB must never exceed $(F_{\max} - F_{\min})/2^n$. Therefore, the measurement requires

$$\frac{(F_{\max})^2}{N} \times (1 \text{ LSB in } NT) \leq \frac{F_{\max} - F_{\min}}{2^n}$$

270434-3

WILLIAMSON: USING THE 8051 MICROCONTROLLER

The required resolution in NT is, therefore,

$$1 \text{ LSB in } NT \leq \frac{N \times (F_{\max} - F_{\min})}{2^n \times (F_{\max})^2}$$

Now, to say that NT is measured with m -bit resolution means

$$1 \text{ LSB in } NT = \frac{N \times (1/F_{\min} - 1/F_{\max})}{2^m}$$

Substituting this value for 1 LSB into the required resolution and solving for 2^m yields

$$2^m \geq \frac{F_{\max}}{F_{\min}} \times 2^n$$

Then the requirement on m is

$$m \geq n + \frac{\ln (F_{\max}/F_{\min})}{\ln (2)}$$

It can be stated with some certainty, then, that if the transducer frequency varies over a range of 2-to-1, the frequency can be calculated with 8-bit resolution from a period measurement that has 9-bit resolution. If the frequency variation is less than 2-to-1, another full bit of resolution in the period measurement is not needed.

To obtain m -bit resolution in NT , N must satisfy

$$N \geq \frac{2^m}{T_{\max} - T_{\min}}$$

Substituting for 2^m , and using $T_{\max} = 1/F_{\min}$ and $T_{\min} = 1/F_{\max}$, gives the result that

$$N \geq \frac{(F_{\max})^2}{F_{\max} - F_{\min}} \times 2^n$$

It should be noted that the units of frequency here are periods/machine cycle, since the 8051 measures time by counting machine cycles. The conversion factor between Hz and periods/machine cycle is $12/(\text{clock frequency})$. So the requirement on N can also be written

$$N \geq \frac{F_{\max}}{F_{\max} - F_{\min}} \times \frac{F_{\max}}{F_{\text{xtal}}} \times 12 \times 2^n$$

where F_{xtal} is the 8051 clock frequency in the same units as F_{\max} and F_{\min} . This is the number of transducer pulses over which the transducer signal must be sampled to achieve the required resolution in F .

For example, suppose that 8-bit resolution is required in F , where $F_{\max} = 10 \text{ kHz}$ and $F_{\min} = 5 \text{ kHz}$, and that $F_{\text{xtal}} = 12 \text{ MHz}$. Then the above calculation shows that $N = 6$ periods gives sufficient resolution in the period measurement to satisfy the resolution requirement in F . Six periods will take 0.6–1.2 ms of sampling time, on that frequency range. Recall that the sample time for a direct frequency measurement of the same signal and to the same resolution was earlier calculated to be 51.2 ms.

X. COMPUTING THE FREQUENCY FROM THE PERIOD

The period measurement leaves one with a 16-bit integer, which is $N \times T$ (or NT) in machine cycles. The conversion to frequency is straightforward:

$$F = N/(NT) \text{ periods/machine cycle.}$$

The quantity of interest is probably not F , but a normalized measure of the amount by which F exceeds its minimum acceptable value. This quantity represents, through the transducer's transfer function, the "reported value" of the measurand, and this quantity is an n -bit integer whose value ranges from 0 (all bits = 0) to full scale (all bits = 1). This normalized frequency is

$$f = \frac{F - F_{\min}}{F_{\max} - F_{\min}} = \frac{F_{\min}}{F_{\max} - F_{\min}} \times (F/F_{\min} - 1)$$

Using $F = N/(NT)$ and $F_{\min} = N/(NT_{\max})$ allows the normalized frequency to be written

$$f = \frac{F_{\min}}{F_{\max} - F_{\min}} \times \frac{NT_{\max} - NT}{NT}$$

To get a handle on what kinds of numbers are involved here, consider the situation where 8-bit resolution is required in f , and in which $F_{\text{xtal}} = 12 \text{ MHz}$, $F_{\max} = 10 \text{ kHz}$, and $F_{\min} = 5 \text{ kHz}$. We have previously determined that for this set of conditions, $N = 6$ periods gives sufficient resolution in the period measurement to satisfy the resolution requirement in F (and in f). With a 12 MHz clock frequency, T_{\max} in machine cycles is $(12\,000 \text{ kHz})/(12 \times 5 \text{ kHz}) = 200$ machine cycles, so NT_{\max} is $6 \times 200 = 1200$ machine cycles. The calculation for f then becomes

$$f = \frac{1200 - NT}{NT}$$

The minimum acceptable value that NT can have is $(N \times T_{\min} + 1)$, where $T_{\min} = (12\,000 \text{ kHz})/(12 \times 10 \text{ kHz}) = 100$ machine cycles. Then $N \times T_{\min} = 6 \times 100 = 600$ machine cycles. The allowable values for NT are then 601–1200 machine cycles, a total of 599 different values.

The fastest way to "calculate" f would be with a 599-byte look-up table. This method has the added advantage that nonlinearities in the transfer function between frequency and measurand can be built into the table. Look-up tables are facilitated in the 8051 by the `MOVC A, @A + PC`, and `MOVC A, @A + DPTR` instructions. `DPTR` is a 16-bit "data pointer" register in the 8051. Its two bytes can be individually addressed as `DPL` (low byte) and `DPH` (high byte).

In the example under discussion, it will be necessary to load `DPTR` with the address of the first byte of the look-up table, less 601, plus the 2-byte value of NT . The software that accesses the table might then take the following form:

TABLE EQU (address of first table entry)

270434–4

DIVIDE ROUTINE

This routine calculates the quotient and remainder of the division of the numerator by the denominator. The conversion is done by a 16-bit integer.

quotient = numerator / denominator

remainder = numerator - quotient * denominator

If the numerator and denominator are integers and the denominator is not zero, the quotient is of the form $q_n q_{n-1} q_{n-2} \dots q_1 q_0$ where q_n is the coefficient of 2^{n-1} . The procedure is as follows:

Substituting this value for q_n into the required resolution and solving for q_{n-1} yields

if numerator > denominator then $q_n = 0$ else $q_n = 1$

if $q_n > 0$ then numerator = 2 * numerator

else numerator = 2 * numerator - denominator

increment n

end_while

Fig. 1. A divide algorithm.

```

MOV    A,#LOW(TABLE-601)
ADD    A,NTLO
MOV    DPL,A
MOV    A,#HIGH(TABLE-601)
ADDC   A,NTLH
MOV    DPH,A
CLR    A
MOVC   A,@A+DPTR

```

At this point the accumulator contains the 8-bit value of f .

It is perfectly reasonable to decide that a 599-byte look-up table is unwieldy. Its advantages are speed and built-in error correction. But a reasonably fast divide algorithm can be written to this specific purpose, making use of *a priori* knowledge about the sizes of the numbers that are involved in the computation. It helps to know that in this example the numerator is never going to be larger than 599 and the denominator is always greater than the numerator.

A complete discussion of divide routines is beyond the scope of this paper, but a suitable divide algorithm for this specific application is shown in Fig. 1. Reference [1] calls this the Restoring division algorithm. It is particularly well suited to the 8051, because "<" comparisons are greatly facilitated by the 8051's CJNE (compare and jump if not equal) instruction. CJNE A,B,rel, executes a relative jump if A does not equal B. More importantly to this application, the instruction sets the carry flag if $A < B$.

XI. ACCURACY AND RESOLUTION

The accuracy with which the 8051 will measure the frequency or period of the transducer signal depends on two things: the accuracy of the clock oscillator and variations in the interrupt response time.

The accuracy of the clock oscillator is a function of the crystal used. The 8051 has a built-in oscillator with a 10% tolerance. The accuracy of the interrupt response time is a function of the interrupt source. The 8051 has a built-in interrupt source with a 10% tolerance. The accuracy of the interrupt response time is a function of the interrupt source. The 8051 has a built-in interrupt source with a 10% tolerance.

Since the clock signal is normally generated by a crystal oscillator, the oscillator accuracy normally far exceeds the quantizing error inherent in the finite (n -bit) resolution.

As was previously mentioned, interrupt response time does not introduce an error into the measurement itself, but variations in the interrupt response time can. Interrupt response time in the 8051 can vary from 3 to 8 machine cycles, depending on what instruction is in progress at the time the interrupt is generated. This would represent an error of ± 5 counts in the measured value of NT during a period measurement. An error of ± 5 counts in NT does not necessarily translate to ± 5 LSB's in the final result, but it might still represent an error that exceeds the resolution.

In a direct frequency measurement variations in the interrupt response time would represent an error of $\pm 5 \mu s$ in the sample time.

If these kinds of errors are unacceptable there are ways to deal with them. In period measurements, if the duty cycle of the transducer is constant, the pulswidth measurement technique, previously described, can be used. Its advantage is that it gates the timer off when the interrupt is generated, rather than when the interrupt is responded to.

In other cases one can simply increase the sample time above the minimum required to obtain the desired resolution. For example, if the measurement requires 8-bit resolution, one can design the software for an 11-bit resolution and truncate the result to 8 bits.

REFERENCES

- [1] Davio et al., *Digital Systems with Algorithm Implementation*, New York: Wiley, 1983.

8051 SOFTWARE PACKAGES

- **Choice of hosts:**
PCDOS 3.0 based IBM® PC XT/AT*,
iRMX®86, iPDSTM System, Series II,
Series III, and Series IV
- **Supports all members of the Intel
MCS® -51 architecture**

PL/M51 Software Package Contains the following:

- **PL/M51 Compiler which is designed to support all phases of software implementation**
- **RL51 Linker and Relocator which enables programmers to develop software in a modular fashion**

- **LIB51 Librarian which lets programmers create and maintain libraries of software object modules**

8051 Software Development Package Contains the following:

- **8051 Macro Assembler which gives symbolic access to 8051 hardware features**
- **RL51 Linker and Relocator program which links modules generated by the assembler**
- **LIB51 Librarian which lets programmers create and maintain libraries of software object modules**

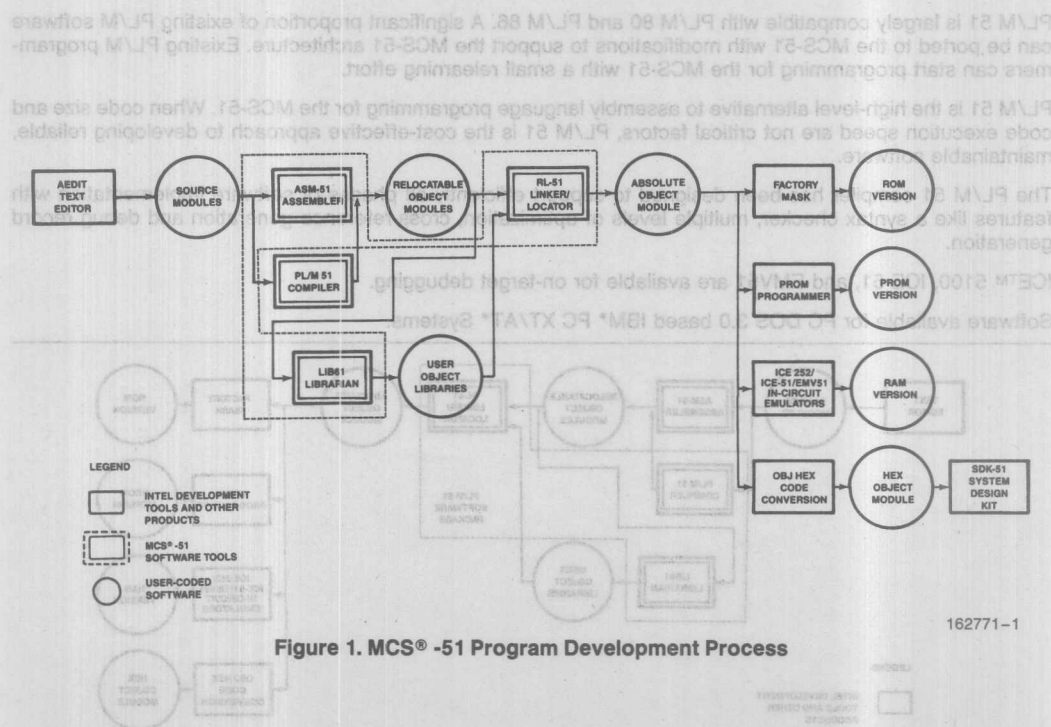


Figure 1. MCS® -51 Program Development Process

162771-1

*IBM and AT are registered trademarks of International Business Machine Corporation.

int 1

- High-level programming language for the Intel MCS® -51 single-chip microcomputer family
- Compatible with PL/M 80 assuring MCS® -80/85 design portability
- Enhanced to support boolean processing
- Tailored to provide an optimum balance among on-chip RAM usage, code size and code execution time
- Produces relocatable object code which is linkable to object modules generated by all other 8051 translators
- Allows programmer to have complete control of microcomputer resources
- Extends high-level language programming advantages to microcontroller software development
- Improved reliability, lower maintenance costs, increased programmer productivity and software portability
- Includes the linking and relocating utility and the library manager
- Supports all members of the Intel MCS® -51 architecture

PL/M 51 is a structured, high-level programming language for the Intel MCS-51 family of microcomputers. The PL/M 51 language and compiler have been designed to support the unique software development requirements of the single-chip microcomputer environment. The PL/M language has been enhanced to support Boolean processing and efficient access to the microcomputer functions. New compiler controls allow the programmer complete control over what microcomputer resources are used by PL/M programs.

PL/M 51 is largely compatible with PL/M 80 and PL/M 86. A significant proportion of existing PL/M software can be ported to the MCS-51 with modifications to support the MCS-51 architecture. Existing PL/M programmers can start programming for the MCS-51 with a small relearning effort.

PL/M 51 is the high-level alternative to assembly language programming for the MCS-51. When code size and code execution speed are not critical factors, PL/M 51 is the cost-effective approach to developing reliable, maintainable software.

The PL/M 51 compiler has been designed to support efficiently all phases of software implementation with features like a syntax checker, multiple levels of optimization, cross-reference generation and debug record generation.

ICE™ 5100, ICE 51, and EMV51 are available for on-target debugging.

Software available for PC DOS 3.0 based IBM® PC XT/AT® Systems.

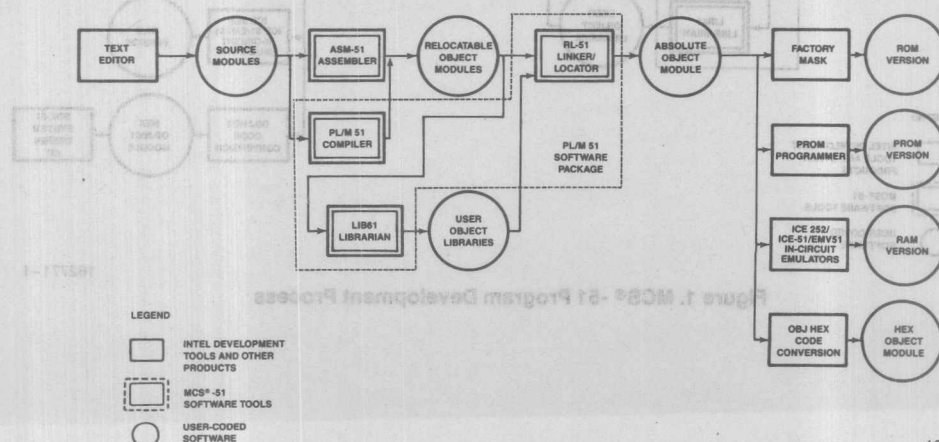


Figure 2. PL/M51 Software Package

162771-2

PL/M 51 COMPILER

FEATURES

Major features of the Intel PL/M 51 compiler and programming language include:

Structured Programming

PL/M source code is developed in a series of modules, procedures, and blocks. Encouraging program modularity in this manner makes programs more readable, and easier to maintain and debug. The language becomes more flexible, by clearly defining the scope of user variables (local to a private procedure, for example).

Language Compatibility

PL/M 51 object modules are compatible with object modules generated by all other MCS-51 translators. This means that PL/M programs may be linked to programs written in any other MCS-51 language.

Object modules are compatible with In-Circuit Emulators and Emulation Vehicles for MCS-51 processors: the DEBUG compiler control provides these tools with symbolic debugging capabilities.

Supports Three Data Types

PL/M makes use of three data types for various applications. These data types range from one to sixteen bits and facilitate various arithmetic, logic, and address functions:

- Bit: a binary digit
- Byte: 8-bit unsigned number or,
- Word: 16-bit unsigned number.

Another powerful facility allows the use of BASED variables that map more than one variable to the same memory location. This is especially useful for passing parameters, relative and absolute addressing, and memory allocation.

Two Data Structuring Facilities

PL/M 51 supports two data structuring facilities. These add flexibility to the referencing of data stored in large groups.

- Array: Indexed list of same type data elements
- Structure: Named collection of same or different type data elements
- Combinations of Both: Arrays of structures or structures of arrays.

Interrupt Handling

A procedure may be defined with the INTERRUPT attribute. The compiler will generate code to save and restore the processor status, for execution of the user-defined interrupt handler routines.

Compiler Controls

The PL/M 51 compiler offers controls that facilitate such features as:

- Including additional PL/M 51 source files from disk
- Cross-reference
- Corresponding assembly language code in the listing file

Program Addressing Control

The PL/M 51 compiler takes full advantage of program addressing with the ROM (SMALL/MEDIUM/LARGE) control. Programs with less than 2 KB code space can use the SMALL or MEDIUM option to generate optimum addressing instructions. Larger programs can address over the full 64 KB range.

Code Optimization

The PL/M 51 compiler offers four levels of optimization for significantly reducing overall program size.

- Combination or "folding" of constant expressions; "Strength reductions" (a shift left rather than multiply by 2)
- Machine code optimizations; elimination of superfluous branches
- Automatic overlaying of on-chip RAM variables
- Register history: an off-chip variable will not be reloaded if its value is available in a register.

Error Checking

The PL/M 51 compiler has a very powerful feature to speed up compilations. If a syntax or program error is detected, the compiler will skip the code generation and optimization passes. This usually yields a 2X performance increase for compilation of programs with errors.

A fully detailed set of programming and compilation error messages is provided by the compiler and user's guide.

BENEFITS

PL/M 51 is designed to be an efficient, cost-effective solution to the special requirements of MCS-51 Microsystem Software Development, as illustrated by the following benefits of PL/M use:

Low Learning Effort

PL/M 51 is easy to learn and to use, even for the novice programmer.

Earlier Project Completion

Critical projects are completed much earlier than otherwise possible because PL/M 51, a structured high-level language, increases programmer productivity.

Lower Development Cost

Increases in programmer productivity translate immediately into lower software development costs because less programming resources are required for a given programmed function.

Increased Reliability

PL/M 51 is designed to aid in the development of reliable software (PL/M programs are simple statements of the program algorithm). This substantially reduces the risk of costly correction of errors in systems that have already reached full production status, as the more simply stated the program is, the more likely it is to perform its intended function.

Easier Enhancements and Maintenance

Programs written in PL/M tend to be self-documenting, thus easier to read and understand. This means it is easier to enhance and maintain PL/M programs as the system capabilities expand and future products are developed.

RL51 LINKER AND RELOCATOR

- Links modules generated by the assembler and the PL/M compiler
- Locates the linked object to absolute memory locations

- Enables modular programming of software-efficient program development
- Modular programs are easy to understand, maintainable and reliable

The MCS-51 linker and relocater (RL51) is a utility which enables MCS-51 programmers to develop software in a modular fashion. The utility resolves all references between modules and assigns absolute memory locations to all the relocatable segments, combining relocatable partial segments with the same name.

With this utility, software can be developed more quickly because small functional modules are easier to understand, design and test than large programs.

The total number of allowed symbols in user-developed software is very large because the assembler number of symbols' limit applies only per module, not to the entire program. Therefore programs can be more readable and better documented. RL51 can be invoked either manually or through a batch file for improved productivity.

Modules can be saved and used on different programs. Therefore the software investment of the customer is maintained.

RL51 produces two files. The absolute object module file can be directly executed by the MCS-51 family. The listing file shows the results of the link/locate process.

LIB51 LIBRARIAN

The LIB51 utility enables MCS-51 programmers to create and maintain libraries of software object modules. With this utility, the customer can develop standard software modules and place them in libraries, which programs can access through a standard interface. When using object libraries, the linker will

call only object modules that are required to satisfy external references.

Consequently, the librarian enables the customer to port and reuse software on different projects—thereby maintaining the customer's software investment.

ORDERING INFORMATION

Order Code

D86PLM51

R86PLM51

Operating Environment

PL/M51 Software for PC DOS 3.0 Systems

PL/M51 Software for iRMX 86 Systems

Documentation Package

PL/M 51 User's Guide

MCS-51 Utilities User's Guide

SUPPORT:

Hotline Telephone Support, Software Performance Report (SPR), Software Updates, Technical Reports, and monthly Technical Newsletters are available.

- Symbolic relocatable assembly language programming for 8051 microcontrollers
- Extends Intellec® Microcomputer Development System to support 8051 program development
- Produces Relocatable Object Code which is linkable to other 8051 Object Modules

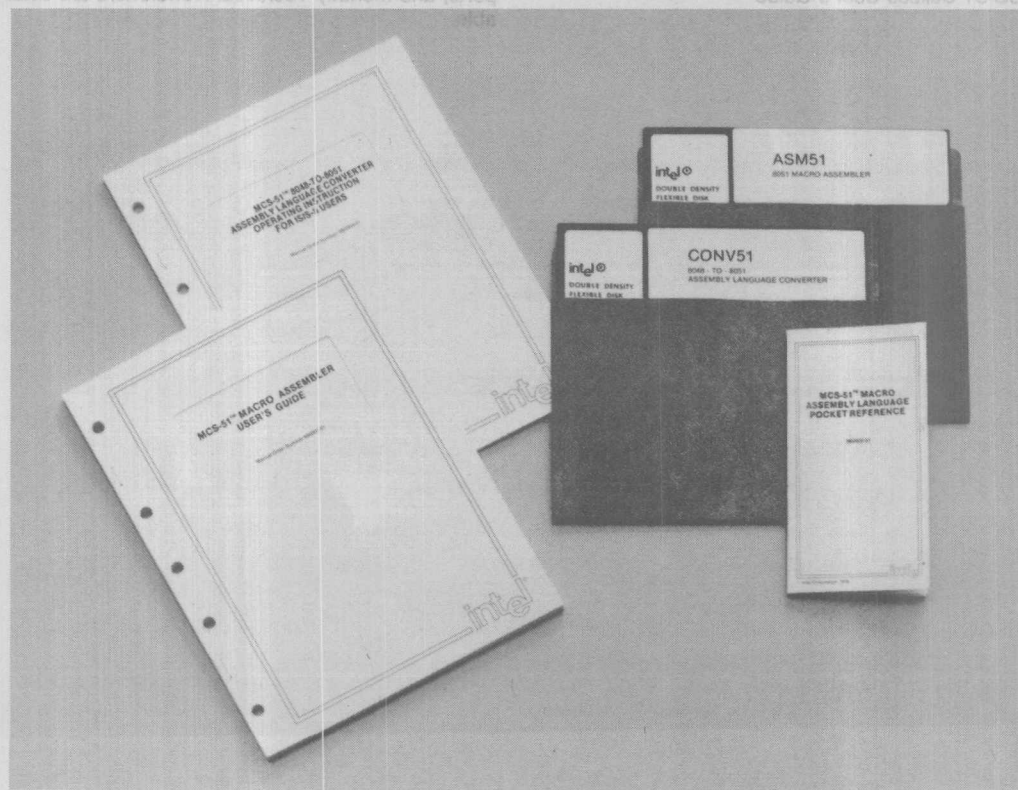
- Encourage modular program design for maintainability and reliability
- Macro Assembler features conditional assembly and macro capabilities
- Supports all members of the Intel MCS® 51 architecture

The 8051 software development package provides development system support for the powerful 8051 family of single chip microcomputers. The package contains a symbolic macro assembler and relocation/linkage utilities.

The assembler produces relocatable object modules from 8051 macro assembly language instructions. The object code modules can be linked and located to absolute memory locations. This absolute object code may be used to program the 8751 EPROM version of the chip. The assembler output may also be debugged using the new family of ICE 5100 emulators or with the ICE-51™ in-circuit emulator.

The converter translates 8048 assembly language instructions into 8051 source instructions to provide software compatibility between the two families of microcontrollers.

Software available for PC DOS 3.0 based IBM® PC XT/AT Systems.



162771-3

■ Supports 8051 family program development on Inteltec® Microcomputer Development Systems

■ Gives symbolic access to powerful 8051 hardware features

■ Produces object file, listing file and error diagnostics

■ Object files are linkable and locatable

■ Provides software support for many addressing and data allocation capabilities

■ Symbolic Assembler supports symbol table, cross-reference, macro capabilities, and conditional assembly

The 8051 Macro Assembler (ASM51) translates symbolic 8051 macro assembly language modules into linkable and locatable object code modules. Assembly language mnemonics are easier to program and are more readable than binary or hexadecimal machine instructions. By allowing the programmer to give symbolic names to memory locations rather than absolute addresses, software design and debug are performed more quickly and reliably. Furthermore, since modules are linkable and relocatable, the programmer can do his software in modular fashion. This makes programs easy to understand, maintainable and reliable.

The assembler supports macro definitions and calls. This is a convenient way to program a frequently used code sequence only once. The assembler also provides conditional assembly capabilities.

Cross referencing is provided in the symbol table listing, showing the user the lines in which each symbol was defined and referenced.

ASM51 provides symbolic access to the many useful addressing features of the 8051 architecture. These features include referencing for bit and byte locations, and for providing 4-bit operations for BCD arithmetic. The assembler also provides symbolic access to hardware registers, I/O ports, control bits, and RAM addresses. ASM51 can support all members of the 8051 family.

Math routines are enhanced by the MULTIPLY and DIVIDE instructions.

If an 8051 program contains errors, the assembler provides a comprehensive set of error diagnostics, which are included in the assembly listing or on another file. Program testing may be performed by using the iUP Universal Programmer and iUP F87/51 personality module to program the 8751 EPROM version of the chip.

ICE 5100, ICE51 and EMV51 are available for program debugging.

RL51 LINKER AND RELOCATOR PROGRAM

■ Links modules generated by the assembler

■ Locates the linked object to absolute memory locations

■ Enables modular programming of software for efficient program development

■ Modular programs are easy to understand, maintainable and reliable

The 8051 linker and relocator (RL51) is a utility which enables 8051 programmers to develop software in a modular fashion. The linker resolves all references between modules and the relocator assigns absolute memory locations to all the relocatable segments, combining relocatable partial segments with the same name.

With this utility, software can be developed more quickly because small functional modules are easier to understand, design and test than large programs.

The number of symbols in the software is very large because the assembler symbol limit applies only per module not the entire program. Therefore programs can be more readable and better documented.

Modules can be saved and used on different programs. Therefore the software investment of the customer is maintained.

RL51 produces two files. The absolute object module file can be directly executed by the 8051 family. The listing file shows the results of the link/locate process.

LIB51 LIBRARIAN

The LIB51 utility enables MCS-51 programmers to create and maintain libraries of software object modules. With this utility, the customer can develop standard software modules and place them in libraries, which programs can access through a standard interface. When using object libraries, the linker will call only object modules that are required to satisfy external references.

Consequently, the librarian enables the customer to port and reuse software on different projects—thereby maintaining the customer's software investment.

ORDERING INFORMATION

Order Code	Operating Environment
D86ASM51	8051 Assembler for PC DOS 3.0 Systems
R86ASM51	8051 Assembler for iRMX 86 Systems

Documentation Package:

MCS-51 Macro Assembler User's Guide

MCS-51 Utilities User's Guide for 8080/8085
Based Development System

MCS-51 8048-to-8051 Assembly Language Con-
verter Operating Instructions for ISIS-II Users

SUPPORT:

Hotline Telephone Support, Software Performance
Reporting (SPR), Software Updates, Technical Re-
ports, Monthly Newsletter available.

ICETM-5100/252 In-Circuit Emulator for the MCS®-51 Family of Microcontrollers

- **Precise, Full-Speed, Real-Time Emulation of Selected MCS-51 Microcontroller Components at Speeds Up to and Including 16 MHz**
- **64 KB of Mappable High-Speed Emulation Memory**
- **254 24-Bit Frames of Trace Memory (16 Bits Trace Program Execution Addresses and 8 Bits Trace External Events)**
- **Serial Link to the IBM* PC AT, PC XT (and DOS Compatibles), and the Intellec® Series III/IV**
- **ASM-51 and PL/M-51 Language Support**
- **Symbolic Debugging Enables Access to Memory Locations and Program Variables**
- **Four Address Breakpoints with In-Range, Out-Of-Range, and Page Breaks**
- **Equipped with the Integrated Command Directory (ICD™) that Includes:**
 - On-Line Help
 - Syntax Guidance and Checking
 - Dynamic Command-Entry
 - Error Checking
 - Command Recall
- **On-Line Disassembler and Single-Line Assembler to Help with Code Patching**
- **Built-In CRT-Oriented Text Editor**

The ICETM-5100/252 in-circuit emulator is a high-level, interactive debugging system that is used to develop and test the hardware and software of a target system based on the MCS®-51 family of microcontrollers. The ICE-5100/252 emulator can be serially linked to an IBM PC AT or PC XT, or an Intellec Series III/IV. The emulator can communicate with the host system at standard baud rates up to 19.2K. The design of the emulator supports selected MCS-51 microcontroller components at speeds up to and including 16 MHz.



*IBM is a registered trademark of International Business Machines Corporation.

280200-1

The ICE-5100/252 emulator provides full emulation support for the MCS®-51 family members listed in Table 1.

The ICE-5100/252 emulator enables hardware and software development to proceed simultaneously. With the ICE-5100/252 emulator, prototype hardware can be added to the system as it is designed and software can be developed prior to the completion of the hardware prototype. Software and hardware integration can occur while the product is being developed.

The ICE-5100/252 emulator assists four stages of development:

- Software debugging
- Hardware debugging
- System integration
- System test

Software Debugging

The ICE-5100/252 emulator can be operated without being connected to the target system or before any of the user's hardware is available (provided external data RAM is not needed). In this stand-alone mode, the ICE-5100/252 emulator can be used to facilitate program development.

Hardware Debugging

The ICE-5100/252 emulator's AC/DC parametric characteristics match the microcontroller's. The emulator's full-speed operation makes it a valuable tool for debugging hardware, including time-critical serial port, timer, and external interrupt interfaces.

Integration of software and hardware can begin when the emulator is plugged into the microcontroller socket of the prototype system hardware. Hardware can be added, modified, and tested immediately. As each section of the user's hardware is completed, it can be added to the prototype. Thus, the hardware and software can be system tested in real-time operation as each section becomes available.

System Test

When the prototype is complete, it is tested with the final version of the system software. The ICE-5100/252 emulator is then used for real-time emulation of the microcontroller to debug the system as a completed unit.

The final product verification test can be performed using the ROM or EPROM version of the microcontroller. Thus, the ICE-5100/252 emulator provides the ability to debug a prototype or production system at any stage in its development without introducing extraneous hardware or software test tools.

PHYSICAL DESCRIPTION

The ICE-5100/252 emulator consists of the following components (see Figure 1):

- Power supply
- AC and DC power cables
- Controller pod
- Serial cable (host-specific)
- User probe assembly (consisting of the processor module and the user cable)
- Crystal power accessory (CPA)

Table 1. MCS®-51 Family Support Offered by the ICETM-5100/252 Emulator

Part	On-Chip Program Memory	On-Chip Data Memory
8031	None	128 bytes
80C31	None	128 bytes
8032	None	256 bytes
8051	4 KB-ROM	128 bytes
80C51	4 KB-ROM	128 bytes
8052	8 KB-ROM	256 bytes
80C252	None	256 bytes
83C252	8 KB-ROM	256 bytes
8751	4 KB-EPROM	128 bytes
87C51	4 KB-EPROM	128 bytes
8752	8 KB-EPROM	256 bytes
87C252	8 KB-EPROM	256 bytes

- 40-pin DIP target adaptor
- Clips assembly
- Software (includes the ICE-5100/252 emulator software, diagnostic software, and tutorial)

The controller pod contains 64 KB of emulation memory, 254- by 24-bit frames of trace memory, and the control processor. In addition, the controller pod houses a BNC connector that can be used to connect up to 10 multi-ICE compatible emulators together for synchronous starting and stopping of emulation.

The serial cable connects the host system to the controller pod. The serial cable supports a subset of the RS-232C signals.

The user probe assembly consists of a user cable and a processor module. The processor module houses the emulation processor and the interface logic. The target adaptor connects to the processor module and provides an electrical and mechanical interface to the target microcontroller socket.

The crystal power accessory (CPA) is a small detachable board that connects to the controller pod and enables the ICE-5100/252 emulator to run in

stand-alone mode. The target adaptor plugs into the socket on the CPA; the CPA then supplies clock and power to the user probe.

The clips assembly enables the user to trace external events. Eight bits of data are gathered on the rising edge of PSEN during opcode fetches. The clips information can be displayed using the CLIPS option with the PRINT command. Trace qualification input and output lines are also provided on the clips pod for connection to test equipment.

The ICE-5100/252 emulator software supports mnemonics, object file formats, and symbolic references generated by Intel's ASM-51 and PL/M-51 programming languages. Along with the ICE-5100/252 emulator software is a customer confidence test disk with diagnostic routines that check the operation of the hardware.

The on-line tutorial is written in the ICE-5100 command language. Thus, the user is able to interact with and use the ICE-5100/252 emulator while executing the tutorial.

A comprehensive set of documentation is included with the ICE-5100/252 emulator.

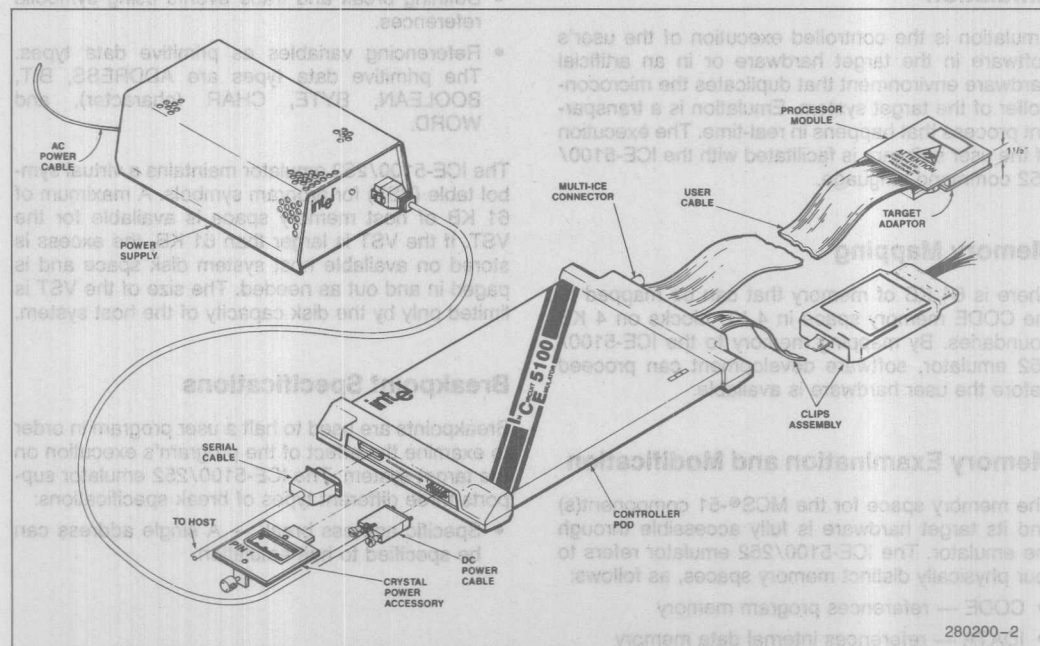


Figure 1. The ICETM-5100/252 System Hardware

ICETM-5100/252 EMULATOR FEATURES

The ICE-5100/252 emulator has been created to assist a product designer in developing, debugging, and testing designs incorporating the MCS®-51 family of microcontrollers. The following sections detail some of the ICE-5100/252 emulator features.

Processor Selection

The ICE-5100/252 emulator emulates the microcontrollers listed in Table 1. Selecting a processor type changes the following characteristics to match the microcontroller selected:

- Internal RAM size
- Internal ROM size
- Idle and power down mode enable
- Special function register symbolic map
- Memory map
- Latched or unlatched \overline{EA}
- Serial port framing and error detection

Emulation

Emulation is the controlled execution of the user's software in the target hardware or in an artificial hardware environment that duplicates the microcontroller of the target system. Emulation is a transparent process that happens in real-time. The execution of the user software is facilitated with the ICE-5100/252 command language.

Memory Mapping

There is 64 KB of memory that can be mapped to the CODE memory space in 4 KB blocks on 4 KB boundaries. By mapping memory to the ICE-5100/252 emulator, software development can proceed before the user hardware is available.

Memory Examination and Modification

The memory space for the MCS®-51 component(s) and its target hardware is fully accessible through the emulator. The ICE-5100/252 emulator refers to four physically distinct memory spaces, as follows:

- CODE — references program memory
- IDATA — references internal data memory
- RDATA — references special function register memory
- XDATA — references external data memory

ICE-5100/252 emulator commands that access memory use one of the special prefixes (e.g., CODE) to specify the memory space.

The microcontroller's special function registers and register bits can be accessed mnemonically (e.g., DPL, TCON, CY) with the ICE-5100/252 emulator software.

Data can be displayed or modified in one of three bases: hexadecimal, decimal, and binary. Data can also be displayed or modified in one of two formats: ASCII and unsigned integer. Program code can be disassembled and displayed as ASM-51 assembler mnemonics. Code can be modified with standard ASM-51 statements using the built-in single-line assembler.

Symbolic references can be used to specify memory locations. A symbolic reference is a procedure name, line number, program variable, or label in the user program that corresponds to a location.

Some typical symbolic functions include:

- Changing or inspecting the value of a program variable by using the symbolic name to access the memory location.
- Defining break and trace events using symbolic references.
- Referencing variables as primitive data types. The primitive data types are ADDRESS, BIT, BOOLEAN, BYTE, CHAR (character), and WORD.

The ICE-5100/252 emulator maintains a virtual symbol table (VST) for program symbols. A maximum of 61 KB of host memory space is available for the VST. If the VST is larger than 61 KB, the excess is stored on available host system disk space and is paged in and out as needed. The size of the VST is limited only by the disk capacity of the host system.

Breakpoint Specifications

Breakpoints are used to halt a user program in order to examine the effect of the program's execution on the target system. The ICE-5100/252 emulator supports three different types of break specifications:

- Specific address break — A single address can be specified to halt emulation.

- Range break — An arbitrary range of addresses can be specified to halt emulation. Program execution within or, optionally, outside the range halts emulation.
- Page break — Up to 256 page breaks can be specified. A page break is defined as a range of addresses that is 256-bytes long and begins on a 256-byte address boundary.

Break registers are user-defined debug definitions used to create and store breakpoint definitions. Break registers can contain multiple breakpoint definitions and can optionally call debug procedures when emulation halts.

Trace Specifications

Tracing can be triggered using specifications similar to those used for breaking. Normally, the ICE-5100/252 emulator traces program activity while the user program is executing. With a trace specification, tracing can be triggered to occur only when specific conditions are met during execution. Up to 254 24-bit frames of trace information are collected in the buffer during emulation. Sixteen of the 24 bits trace instruction execution addresses, and 8 bits capture external events (CLIPS).

The trace buffer display is similar to an ASM-51 program listing shown in Figure 2. The PRINT command enables the user to selectively display the contents of the trace buffer. The user has the option of displaying the clips information as well as disassembled instructions.

Procedures

Debugging procedures (PROC) are a user-named group of ICE-5100/252 emulator commands that are executed as one command. PROCs enable the user to define several commands in a named block structure. The commands are executed by entering the name of the PROC. The PROC bodies are a simple DO...END construct.

PROCs can simulate missing hardware or software, collect debug information, and execute high-level software patches. PROCs can be copied to text files on disk, then recalled for use in later test sessions. PROCs can also serve as program diagnostics, implementing ICE-5100/252 emulator commands or user-defined definitions for special purposes. PROCs can also be used to set breakpoints.

On-Line Syntax Menu

A special menu, called the Integrated Command Directory (ICD), similar to the one used for the I²C™ system and the VLSiCE-96 emulator, aids in creating syntactically correct command lines. Figure 3 shows an example of the ICD and how it changes to reflect the options available for the GO command.

Help

The HELP command provides ICE-5100/252 emulation command assistance via the host system terminal. On-line HELP is available for the ICE-5100/252 emulator commands shown in Figure 4.

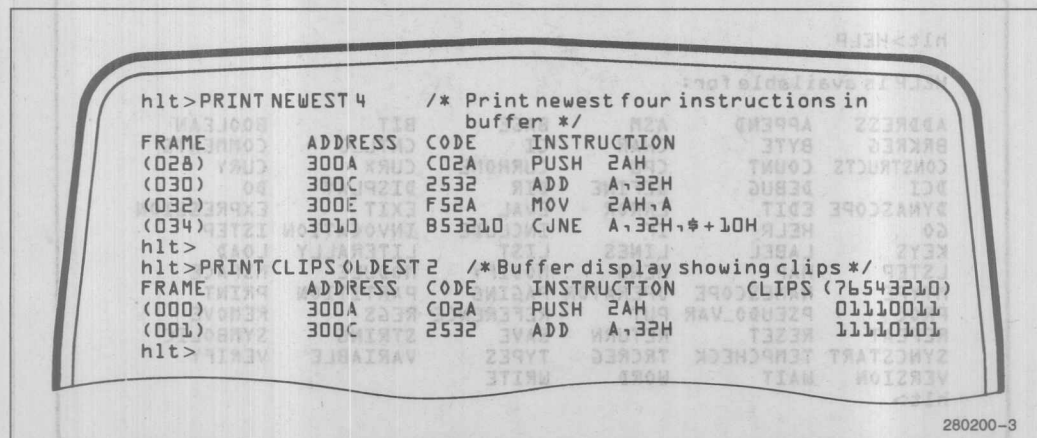


Figure 2. Selected Trace Buffer Displays

The height of the processor module and the target adaptor need to be considered for target systems.

processor module and target adaptor. Figure 5 shows the dimensions of the processor module.

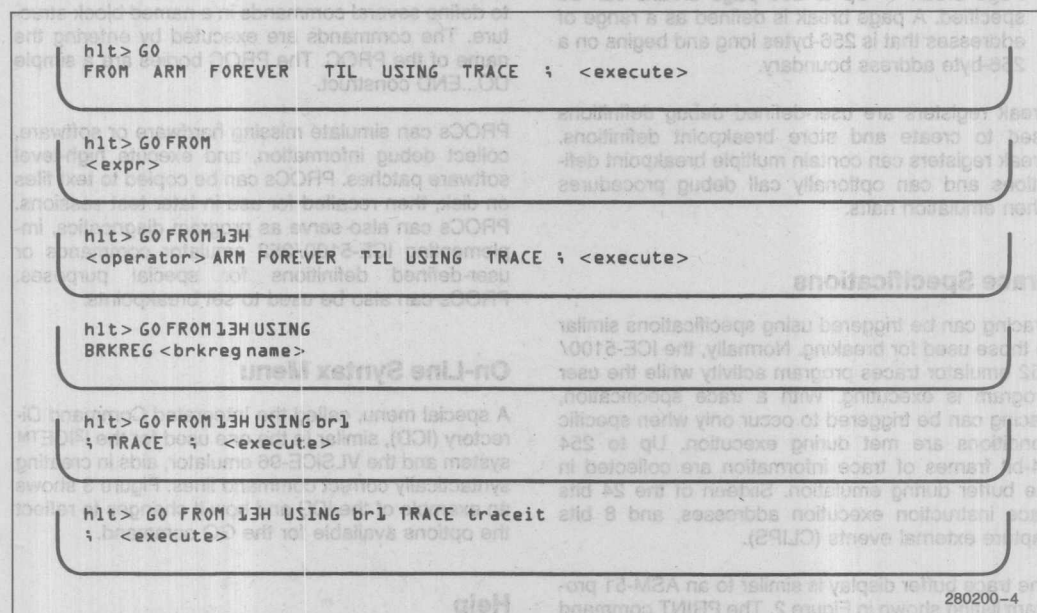


Figure 3. The Integrated Command Directory for the GO Command

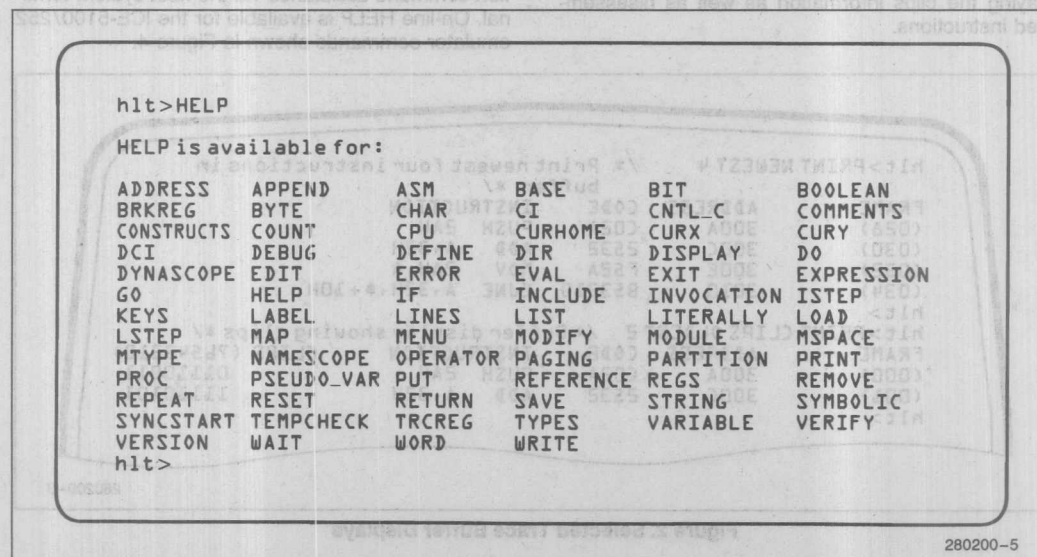


Figure 4. HELP Menu

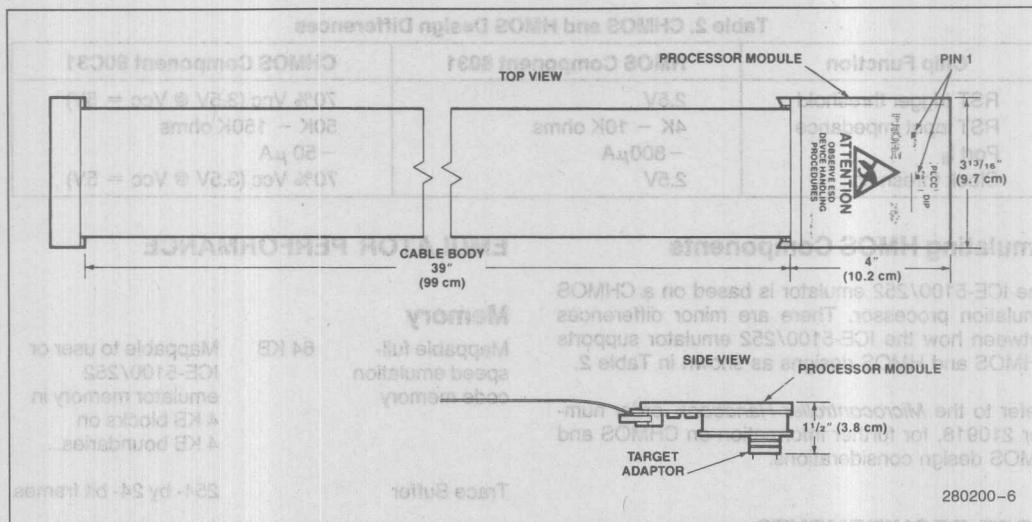


Figure 5. Processor Module Dimensions

ELECTRICAL CONSIDERATIONS

The emulation processor's user-pin timings and loadings are identical to the 80C252 component except as follows.

Maximum Operating ICC and Idle ICC (ma)*

V _{CC}	Maximum Operating ICC (ma)*			Maximum Idle ICC (ma)*		
	4V	5V	6V	4V	5V	6V
Frequency						
0.5 MHz	0.87	1.62	3.0	0.58	1.21	2.5
3.5 MHz	4.8	6.82	9.76	2.2	4.97	6.33
8.0 MHz	10.5	15.0	20.5	6.0	8.98	11.76
12.0 MHz	15.2	22.2	30.2	9.2	13.34	17.46
16.0 MHz	19.4	28.6	38.7	11.8	17.4	23.4

* ICC is measured with all output pins disconnected.
XTAL1 driven with TCLCH, TCHCL = 10ns, V_{il} = V_{ss} + .5V, V_{ih} = V_{cc} - .5V. XTAL2 not connected.

For maximum operating ICC

EA = RST = Port0 = V_{cc}.

For maximum idle ICC

* EA = Port0 = V_{cc}, RST = V_{cc}, internal clock to PCA gated off.

- Up to 25 pf of additional pin capacitance is contributed by the processor module and target adaptor assemblies.
- Pin 31, EA, has approximately 32 pf of additional capacitance loading due to sensing circuitry.
- Pins 18 and 19, XTAL1 and XTAL2, respectively, have approximately 15 to 16 pf of additional capacitance when configured for crystal operation.

Table 2. CHMOS and HMOS Design Differences

Chip Function	HMOS Component 8031	CHMOS Component 80C31
RST trigger threshold	2.5V	70% Vcc (3.5V @ Vcc = 5V)
RST input impedance	4K – 10K ohms	50K – 150K ohms
Port I _{ij}	–800 μ A	–50 μ A
Clock threshold	2.5V	70% Vcc (3.5V @ Vcc = 5V)

Emulating HMOS Components

The ICE-5100/252 emulator is based on a CHMOS emulation processor. There are minor differences between how the ICE-5100/252 emulator supports CHMOS and HMOS designs as shown in Table 2.

Refer to the *Microcontroller Handbook*, order number 210918, for further information on CHMOS and HMOS design considerations.

HOST REQUIREMENTS

- IBM PC AT or PC XT (or PC-DOS compatible) with 512 KB of RAM and a hard disk running under the DOS 3.0 (or later) operating system.
- Intellec Series III/IV Microcomputer Development System running under the ISIS or INDX operating system respectively, with at least 512 KB of application memory resident.

Disk drives — Dual floppy or one hard disk and one floppy drive required.

ICETM-5100/252 SYSTEM SOFTWARE PACKAGE

- ICE-5100/252 emulator software
- ICE-5100/252 confidence tests
- ICE-5100/252 tutorial software

EMULATOR PERFORMANCE

Memory

Mappable full-speed emulation code memory 64 KB Mappable to user or ICE-5100/252 emulator memory in 4 KB blocks on 4 KB boundaries.

Trace Buffer 254- by 24- bit frames

Virtual Symbol Table

A maximum of 61 KB of host memory space is available for the Virtual Symbol Table (VST). The rest of the VST resides on disk and is paged in and out as needed.

PHYSICAL CHARACTERISTICS

Controller Pod

Width 8 1/4" (21 cm)
Height 1 1/2" (3.8 cm)
Depth 13 1/2" (34.3 cm)
Weight 4 lbs (1.85 kg)

User Cable

The user cable is 3 feet (approximately 1 m).

Processor Module

(with the target adaptor attached)

Width 3 13/16" (9.7 cm)
Length 1 1/2" (3.8 cm)
Height 1 1/2" (3.8 cm)

Power Supply

Width 7 $\frac{7}{8}$ " (18.1 cm)
 Height 4" (10.06 cm)
 Depth 11" (27.97 cm)
 Weight 15 lbs (6.1 kg)

Serial Cable

The serial cable is 12 feet (3.6 m).

ELECTRICAL CHARACTERISTICS

Power Supply

100 - 120V or 200 - 240V (selectable)
 50 - 60 Hz
 2 amps (AC max) @ 120V
 1 amp (AC max) @ 240V

ENVIRONMENTAL CHARACTERISTICS

Operating temperature +10° C to +40° C (50°F to 104°F)
 Operating humidity Maximum of 85% relative humidity, non-condensing

ORDERING INFORMATION

Emulator Hardware and Software

Order Code Description

pl252KITAD This kit contains: ICE-5100/252 user probe assembly, power supply and cables, serial cables, target adaptor, CPA, ICE-5100 controller pod, software, and documentation for use with an IBM PC AT or PC XT. The kit also includes the 8051 Software Development Package and the AEDIT text editor for use on DOS systems. [Requires software license.]

pl252KITD This kit is the same as the pl252KITAD kit excluding the 8051 Software Development Package and the AEDIT text editor. [Requires software license.]

pl252KITAS This kit contains the ICE-5100/252 user probe assembly, power supply and cables, serial cables, target adaptor, CPA, ICE-5100 controller pod, software, and documentation for use with Intel hosts (Series III, IV). The kit also includes the 8051 Software Development Package and the AEDIT text editor for use on Series III/IV. [Requires software license.]

pl252KITS This kit is the same as the pl252KITAS kit excluding the 8051 Software Development Package and the AEDIT text editor. [Requires software license.]

Software Only

Order Code Description

pSA252D This kit contains the host, probe, diagnostic and tutorial software on 5 $\frac{1}{4}$ " disks for use on an IBM PC AT or PC XT (requires DOS 3.0 or later). [Requires software license.]

pSA252S This kit contains the host, probe, diagnostic and tutorial software on 8" disks (both single-density and double-density) for use on a Series III, and on 5 $\frac{1}{4}$ " disks for use on a Series IV. [Requires software license.]

Other Useful Intel Debug and Development Support Products

Order Code Description

pD86ASM51 8051 Software Development Package (DOS version) — Consists of the ASM-51 macro assembler which gives symbolic access to 8051 hardware features; the RL51 linker and relocater program that links modules generated by ASM-51; CONV51 which enables software written for the MCS-48 family to be up-graded to run on the 8051, and the LIB51 Librarian which programmers can use to create and maintain libraries of software object modules. Use with the DOS operating system (version 3.0 or later).

version) — Consists of the PL/M-51 compiler which provides high-level programming language support; the LIB51 utility that creates and maintains libraries of software object modules; and the RL51 linker and relocater program that links modules generated by ASM-51 and PL/M-51 and locates the linked object modules to absolute memory locations. Use the DOS operating system (version 3.0 or later).

age (ISIS version) — Same as the pD86ASM51 package except this one is for use with the Series III.

pD86PLM51 PL/M-51 Software Package — Same as the pD86PLM51 package except this one is for use with the Series III and Series IV.

pD86EDIEU AEDIT text editor for use with the DOS operating system.

ELECTRICAL CHARACTERISTICS

Power Supply

100 - 120V or 200 - 240V (selectable)
50 - 60 Hz
2 amp (AC max) @ 120V
1 amp (AC max) @ 240V

ENVIRONMENTAL CHARACTERISTICS

Operating temperature +10°C to +40°C (50°F to 104°F)
Operating humidity Maximum of 85% relative humidity, non-condensing

ORDERING INFORMATION

Emulator Hardware and Software

Order Code Description
pD86KITAD This kit contains ICE-8100/525 user probe assembly, power supply and cables, serial cables, target adaptor, CPU, ICE-8100 controller port, software, and documentation for use with an IBM PC AT or PC XT. The kit also includes the 8081 Software Development Package and the AEDIT text editor for use on DOS systems. [Requires software license.]

pD86KITD This kit is the same as the pD86KITAD kit excluding the 8081 Software Development Package and the AEDIT text editor. [Requires software license.]

Other Useful Intel Debug and Development Support Products

Order Code Description
pD86ASM51 8081 Software Development Package (DOS version) — Consists of the ASM-81 macro assembler which gives symbolic access to 8081 hardware features; the RL51 linker and relocater program that links modules generated by ASM-81; CONVRT which enables software written for the iAPX-86 family to be upgraded to run on the 8081 and the LIB51 utility which programmers can use to create and maintain libraries of software object modules. Use with the DOS operating system (version 3.0 or later).

MCS®-96 ARCHITECTURAL OVERVIEW

There are two groups of parts within the MCS®-96 family: the standard 8X9X parts and the 8X9XBH parts. There are several enhancements on the 8X9XBH parts that are not on the 8X9X parts. This manual is written about the 8X9XBH parts, generically referred to as an 8096BH. Where the standard 8X9X parts differ from the 8096BH, notations will be made.

The 8096BH can be separated into several sections for the purpose of describing its operation. There is a 16-bit CPU, a programmable High Speed I/O Unit, an analog to digital converter, a serial port, and a Pulse Width Modulated (PWM) output for digital to analog conversion. In addition to these functional units, there are some sections which support overall operation of the chip such as the clock generator. The CPU and the programmable I/O make the 8096BH very different from any other microcontroller. Let us first examine the CPU.

1.0 CPU OPERATION

The major components of the CPU on the 8096BH are the Register File and the RALU. Communication with the outside world is done through either the Special Function Registers (SFRs) or the Memory Controller. The RALU (Register/Arithmetic Logic Unit) does not use an accumulator, it operates directly on the 256-byte register space made up of the Register File and the SFRs. Efficient I/O operations are possible by directly controlling the I/O through the SFRs. The main benefits of this structure are the ability to quickly change context, the absence of accumulator bottleneck, and fast throughput and I/O times.

1.1 CPU Buses

A "Control Unit" and two busses connect the Register File and RALU. Figure 1 shows the CPU with its

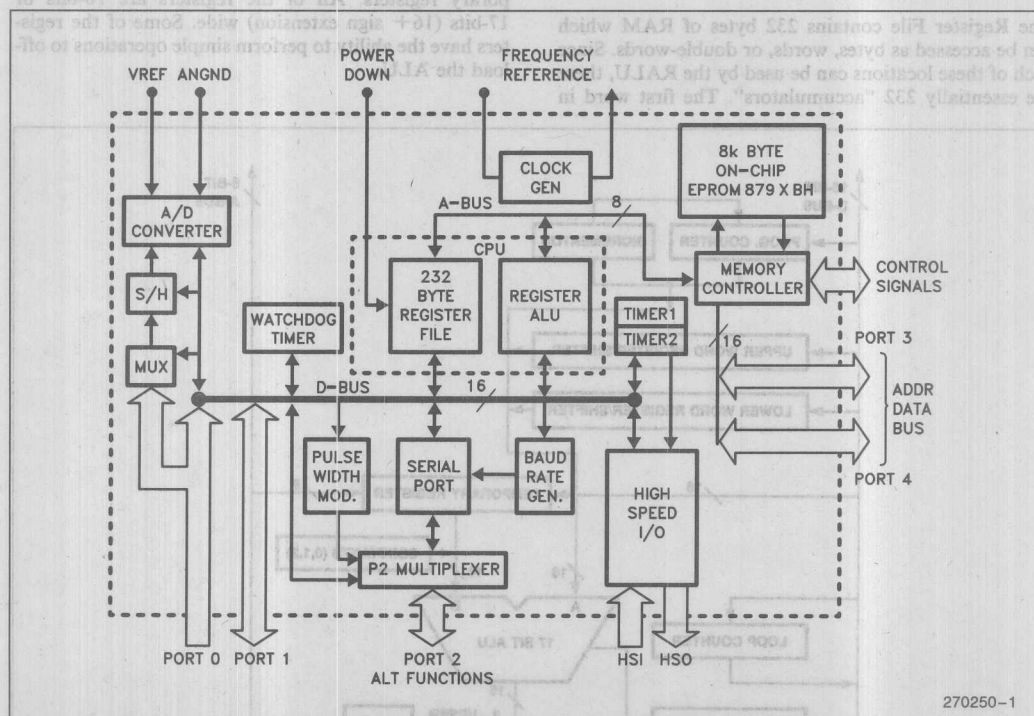


Figure 1. Block Diagram

major bus connections. The two buses are the "A-Bus" which is 8-bits wide, and the "D-Bus" which is 16-bits wide. The D-Bus transfers data only between the RALU and the Register File or Special Function Registers (SFRs). The A-Bus is used as the address bus for the above transfers or as a multiplexed address/data bus connecting to the "Memory Controller". Any accesses of either the internal ROM or external memory are done through the Memory Controller.

Within the memory controller is a slave program counter (Slave PC) which keeps track of the PC in the CPU. By having most program fetches from memory referenced to the slave PC, the processor saves time as addresses seldom have to be sent to the memory controller. If the address jumps sequence then the slave PC is loaded with a new value and processing continues. Data fetches from memory are also done through the memory controller, but the slave PC is bypassed for this operation.

1.2 CPU Register File

The Register File contains 232 bytes of RAM which can be accessed as bytes, words, or double-words. Since each of these locations can be used by the RALU, there are essentially 232 "accumulators". The first word in

the Register File is reserved for use as the stack pointer so it can not be used for data when stack manipulations are taking place. Addresses for accessing the Register File and SFRs are temporarily stored in two 8-bit address registers by the CPU hardware.

1.3 RALU Control

Instructions to the RALU are taken from the A-Bus and stored temporarily in the instruction register. The Control Unit decodes the instructions and generates the correct sequence of signals to have the RALU perform the desired function. Figure 1 shows the instruction register and the control unit.

1.4 RALU

Most calculations performed by the 8096BH take place in the RALU. The RALU, shown in Figure 2, contains a 17-bit ALU, the Program Status Word (PSW), the Program Counter (PC), a loop counter, and three temporary registers. All of the registers are 16-bits or 17-bits (16+ sign extension) wide. Some of the registers have the ability to perform simple operations to off-load the ALU.

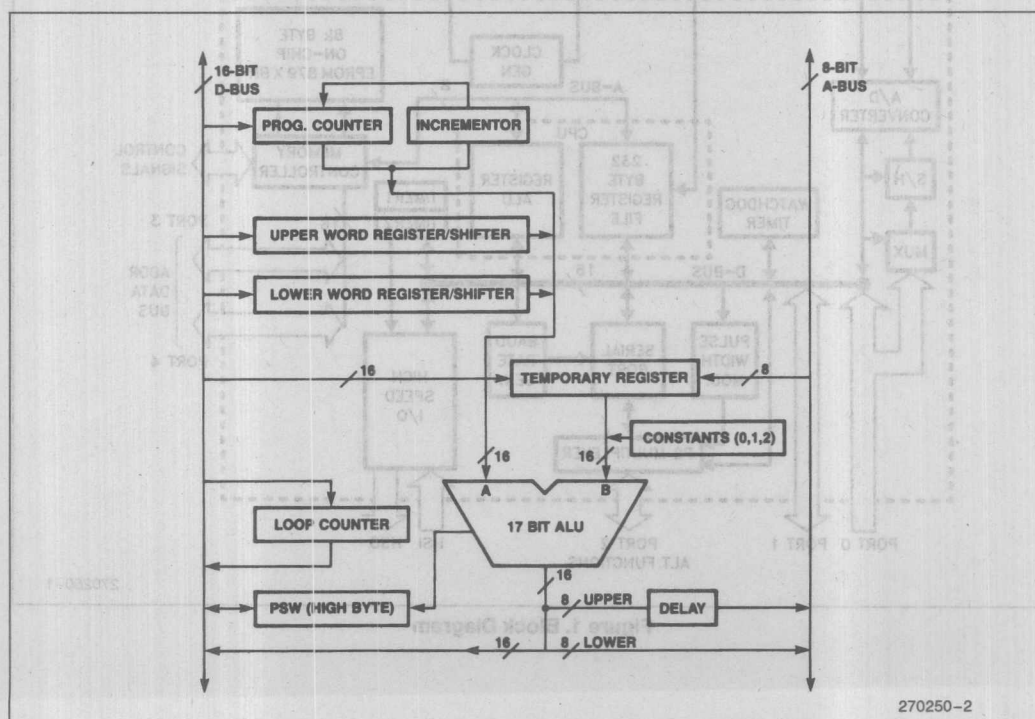


Figure 2. RALU Block Diagram

A separate incrementor is used for the PC; however, jumps must be handled through the ALU. Two of the temporary registers have their own shift logic. These registers are used for the operations which require logical shifts, including Normalize, Multiply, and Divide. The "Lower Word" register is used only when double-word quantities are being shifted, the "Upper Word" register is used whenever a shift is performed or as a temporary register for many instructions. Repetitive shifts are counted by the 5-bit "Loop Counter".

A temporary register is used to store the second operand of two operand instructions. This includes the multiplier during multiplications and the divisor during divisions. To perform subtractions, the output of this register can be complemented before being placed into the "B" input of the ALU.

The DELAY shown in Figure 2 is used to convert the 16-bit bus into an 8-bit bus. This is required as all addresses and instructions are carried on the 8-bit A-Bus. Several constants, such as 0, 1 and 2 are stored in the RALU for use in speeding up certain calculations. These come in handy when the RALU needs to make a 2's complement number or perform an increment or decrement instruction.

1.5 Internal Timing

The 8096BH requires an input clock frequency of between 6.0 MHz and 12 MHz to function. This frequency can be applied directly to XTAL1. Alternatively, since XTAL1 and XTAL2 are inputs and outputs of an inverter, it is also possible to use a crystal to generate the clock. A block diagram of the oscillator section is shown in Figure 3. Details of the circuit and suggestions for its use can be found in Section 1 of the Hardware Design chapter.

The crystal or external oscillator frequency is divided by 3 to generate the three internal timing phases as shown in Figure 4. Each of the internal phases repeat every 3 oscillator periods: 3 oscillator periods are referred to as one "state time", the basic time measurement for 8096BH operations. Most internal operations are synchronized to either Phase A, B or C, each of which have a 33% duty cycle. Phase A is represented externally by CLKOUT, a signal available on the 68-pin part. Phases B and C are not available externally. The relationships of XTAL1, CLKOUT, and Phases A, B, and C are shown in Figure 4. It should be noted that propagation delays have not been taken into account in this diagram. Details on these and other timing relationships can be found in the Hardware Design chapter.

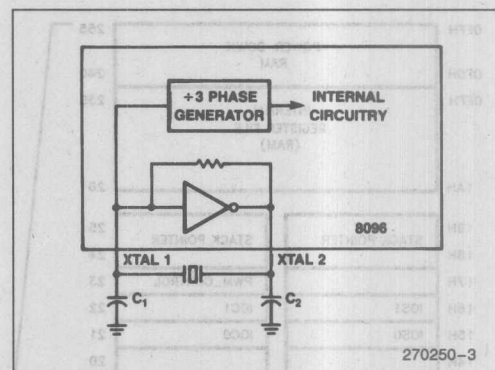


Figure 3. Block Diagram of Oscillator

The RESET line can be used to start the 8096BH at an exact time to provide for synchronization of test equipment and multiple chip systems. Use of this feature is fully explained under RESET, Section 13.

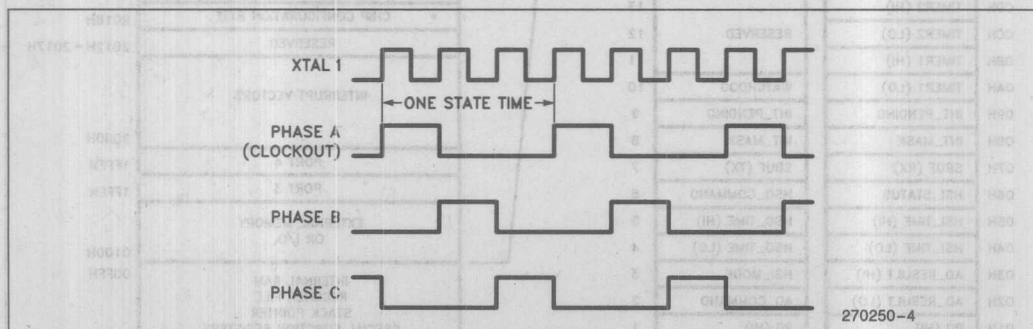


Figure 4. Internal Timings Relative to XTAL 1

2.0 MEMORY SPACE

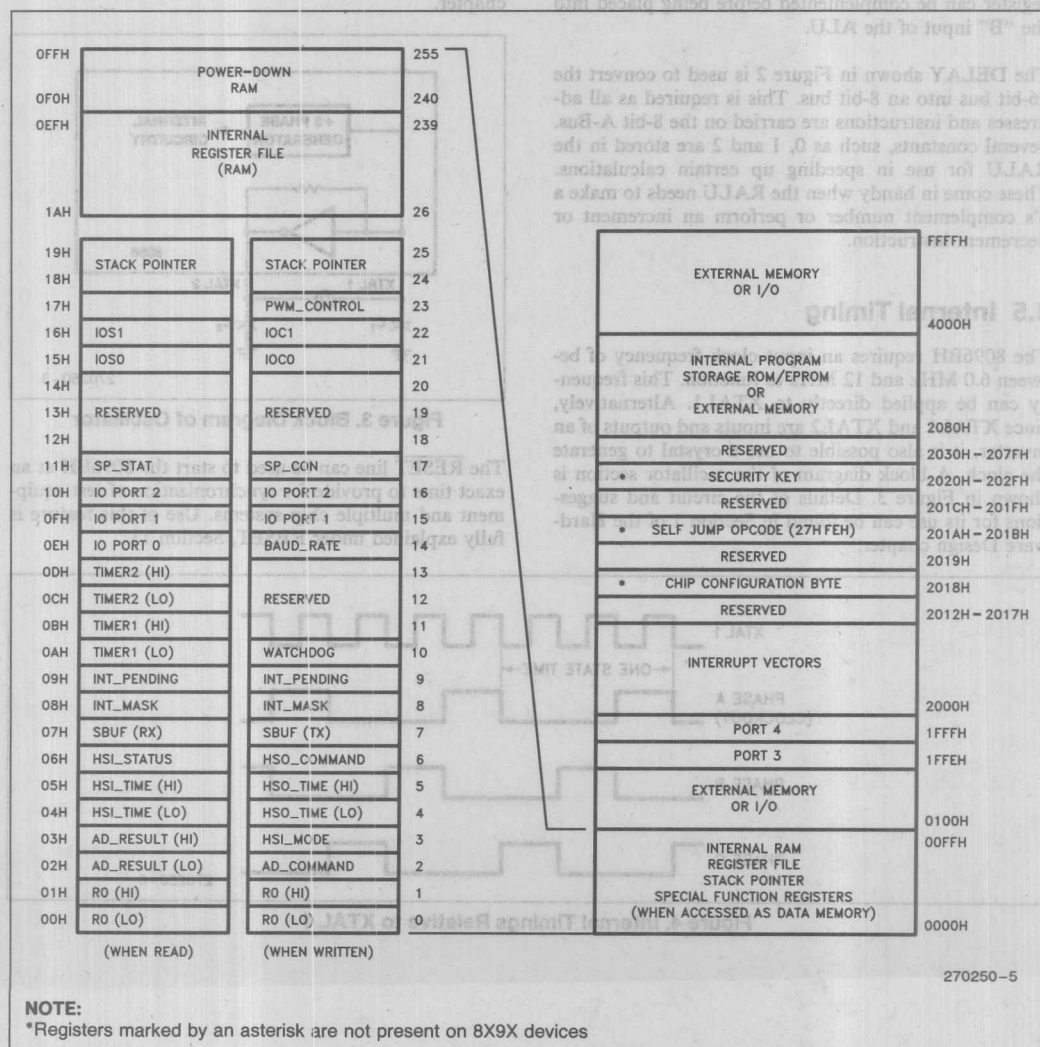
The addressable memory space on the 8096BH consists of 64K bytes, most of which is available to the user for program or data memory. Locations which have special purposes are 0000H through 00FFH and 1FFEH through 2080H. All other locations can be used for either program or data storage or for memory mapped peripherals. A memory map is shown in Figure 5.

2.1 Register File

Locations 00H through 0FFH contain the Register File and Special Function Registers. (SFRs). No code can

be executed from this internal RAM section. If an attempt to execute instructions from locations 000H through 0FFH is made, the instructions will be fetched from *external* memory. This section of external memory is reserved for use by Intel development tools. Execution of a nonmaskable interrupt (NMI) will force a call to external location 0000H, therefore, the NMI instruction is also reserved for Intel development tools.

The RALU can operate on any of the 256 internal register locations. Locations 00H through 17H are used to access the SFRs. Locations 18H and 19H contain the stack pointer. These are not SFRs, and may be used as standard RAM if stack operations are not being performed. The stack pointer must be initialized by the



user program and can point anywhere in the 64K memory space. The stack builds down. There are no restrictions on the use of the remaining 230 locations except that code cannot be executed from them.

2.2 Special Function Registers

All of the I/O on the 8096BH is controlled through the SFRs. Many of these registers serve two functions; one if they are read from, the other if they are written to. Figure 5 shows the locations and names of these registers. A summary of the capabilities of each of these registers is shown in Figure 6, with complete descriptions reserved for later sections.

There are several restrictions on using special function registers.

Neither the source or destination addresses of the Multiply and Divide instructions can be a writable special function register.

These registers may not be used as base or index registers for indirect or indexed instructions.

These registers can only be accessed as bytes unless otherwise specified in Figure 6. Note that some of these registers can only be accessed as words, and not as bytes.

Within the SFR space are several registers labeled "RESERVED". These registers are reserved for future expansion and test purposes. Operations should not be performed with these registers as reads from them and writes to them may produce unexpected results. For example, in some versions of the 8096 writing to location 0CH will set both timers to 0FFFFH. This may not be the case in future products, so it should not be used as a feature.

2.3 Power Down

The upper 16 RAM locations (0F0H through 0FFH) receive their power from the V_{PD} pin. If it is desired to keep the memory in these locations alive during a power down situation, one need only keep voltage on the

V_{PD} pin. The current required to keep the RAM alive is approximately 1 milliamp (refer to the data sheet for the exact specification). Both V_{CC} and V_{PD} must have power applied for normal operation. If V_{PD} is not applied the power down RAM will not function properly, even if V_{CC} is applied.

To place the 8096BH into a power down mode, the \overline{RESET} pin is pulled low. Two state times later the part will be in reset. This is necessary to prevent the part from writing into RAM as the power goes down. The power may now be removed from the V_{CC} pin, the V_{PD} pin must remain within specifications. The 8096BH can remain in this state for any amount of time and the 16 RAM bytes will retain their values.

To bring the 8096BH out of power down, \overline{RESET} is held low while V_{CC} is applied. Two state times after the oscillator has stabilized, the \overline{RESET} pin can be pulled high. *On the 8X9X devices the back-bias generator must also stabilize. This requires approximately 1 millisecond.* The 8096BH will begin to execute code at location 02080H 10 state times after \overline{RESET} is pulled high. Figure 7 shows a timing diagram of the power down sequence. To ensure that the 2 state time minimum reset time (synchronous with CLKOUT) is met, it is recommended that 10 XTAL1 cycles be used. Suggestions for actual hardware connections are given in the Hardware Design Chapter. Reset is discussed in Section 13.

To determine if a reset is a return from power down or a complete cold start a "key" can be written into power-down RAM while the part is running. This key can be checked on reset to determine which type of reset has occurred. In this way the validity of the power-down RAM can be verified. The length of this key determines the probability that this procedure will work, however, there is always a statistical chance that the RAM will power up with a replica of the key.

Under most circumstances, the power-fail indicator which is used to initiate a power-down condition must come from the unfiltered, unregulated section of the power supply. The power supply must have sufficient storage capacity to operate the 8096BH until it has completed its reset operation.

Register	Description	Section
R0	Zero Register — Always reads as a zero, useful for a base when indexing and as a constant for calculations and compares.	3
AD_RESULT	A/D Result Hi/Low — Low and high order Results of the A/D converter (byte read only)	8
AD_COMMAND	A/D Command Register — Controls the A/D	8
HSI_MODE	HSI Mode Register — Sets the mode of the High Speed Input unit.	6
HSI_TIME	HSI Time Hi/Lo — Contains the time at which the High Speed Input unit was triggered. (word read only)	6
HSO_TIME	HSO Time Hi/Lo — Sets the time or count for the High Speed Output to execute the command in the Command Register. (word write only)	7
HSO_COMMAND	HSO Command Register — Determines what will happen at the time loaded into the HSO Time registers.	7
HSI_STATUS	HSI Status Registers — Indicates which HSI pins were detected at the time in the HSI Time registers and the current state of the pins.	6
SBUF (TX)	Transmit buffer for the serial port, holds contents to be outputted.	9
SBUF (RX)	Receive buffer for the serial port, holds the byte just received by the serial port.	9
INT_MASK	Interrupt Mask Register — Enables or disables the individual interrupts.	4
INT_PENDING	Interrupt Pending Register — Indicates that an interrupt signal has occurred on one of the sources and has not been serviced.	4
WATCHDOG	Watchdog Timer Register — Written to periodically to hold off automatic reset every 64K state times.	12
TIMER1	Timer 1 Hi/Lo — Timer 1 high and low bytes. (word read only)	5
TIMER2	Timer 2 Hi/Lo — Timer 2 high and low bytes. (word read only)	5
IOPORT0	Port 0 Register — Levels on pins of port 0.	10
BAUD_RATE	Register which determines the baud rate, this register is loaded sequentially.	9
IOPORT1	Port 1 Register — Used to read or write to Port 1.	10
IOPORT2	Port 2 Register — Used to read or write to Port 2.	10
SP_STAT	Serial Port Status — Indicates the status of the serial port.	9
SP_CON	Serial Port Control — Used to set the mode of the serial port.	9
IOS0	I/O Status Register 0 — Contains information on the HSO status	11
IOS1	I/O Status Register 1 — Contains information on the status of the timers and of the HSI.	11
IOC0	I/O Control Register 0 — Controls alternate functions of HSI pins, Timer 2 reset sources and Timer 2 clock sources.	11
IOC1	I/O Control Register 1 — Controls alternate functions of Port 2 pins, timer interrupts and HSI interrupts.	11
PWM_CONTROL	Pulse Width Modulation Control Register — Sets the duration of the PWM pulse.	8

Figure 6. SFR Summary

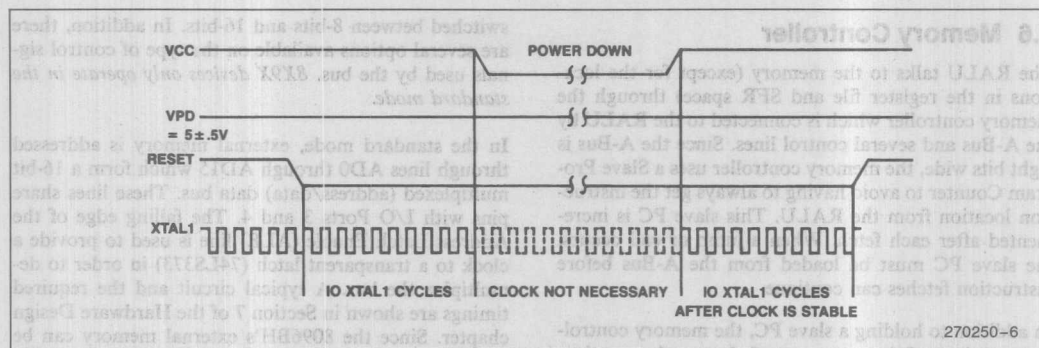


Figure 7. Power Down Timing

2.4 Reserved Memory Spaces

A listing of locations with special significance is shown in Figure 8. The locations marked "Reserved" are reserved by Intel for use in testing or future products. They must be filled with the Hex value FFH to insure compatibility with future parts.

Locations 1FFEh and 1FFFh are reserved for Ports 3 and 4 respectively. This is to allow easy reconstruction of these ports if external memory is used in the system. An example of reconstructing the I/O ports is given in section 7 of the Hardware Design chapter. If ports 3 and 4 are not going to be reconstructed, these locations can be treated as any other external memory location.

The 9 interrupt vectors are stored in locations 2000H through 2011H. The 9th vector is used by Intel development systems, as explained in Section 4.

Locations 2012H through 2017H are reserved for future use. Location 2018H is the Chip Configuration byte which will be discussed in the next section. The Jump-To-Self opcodes at locations 201AH and 201BH are provided for EPROM programming as detailed in the Hardware Design chapter. Locations 2020H through 202FH are the security key used with the ROM Lock feature which will be discussed in the next section. All unspecified addresses in locations 2000H through 207FH, including those marked Reserved, should be considered reserved for use by Intel.

Resetting the 8096BH causes instructions to be fetched starting from location 2080H. This location was chosen to allow a system to have up to 8K of RAM continuous with the register file. Further information on reset can be found in Section 13.

0000H-	0017H	Register Mapped I/O (SFRs)
0018H-	0019H	Stack Pointer
1FFEh-	1FFFh	Ports 3 and 4
2000H-	2011H	Interrupt Vectors
2012H-	2017H	Reserved
2018H		Chip Configuration Byte
2019H		Reserved
201AH-	201BH	"Jump to Self" Opcode (27H FEH)
201CH-	201FH	Reserved
2020H-	202FH	Security Key
2030H-	207FH	Reserved
2080H		Reset Location

Figure 8. Registers with Special Significance

2.5 Internal ROM and EPROM

When a ROM part is ordered, or an EPROM part is programmed, the internal memory locations 2080H through 3FFFh are user specified, as are the interrupt vectors, Chip Configuration Register and Security Key in locations 2000H through 202FH.

Instruction and data fetches from the internal ROM or EPROM occur only if the part has a ROM or EPROM, EA is tied high, and the address is between 2000H and 3FFFh. At all other times data is accessed from either the internal RAM space or external memory and instructions are fetched from external memory. The EA pin is latched on RESET rising. Information on programming EPROMs can be found in Section 10 of the Hardware Design chapter.

Do not execute code out of the last three locations of internal ROM/EPROM.

2.6 Memory Controller

The RALU talks to the memory (except for the locations in the register file and SFR space) through the memory controller which is connected to the RALU by the A-Bus and several control lines. Since the A-Bus is eight bits wide, the memory controller uses a Slave Program Counter to avoid having to always get the instruction location from the RALU. This slave PC is incremented after each fetch. When a jump or call occurs, the slave PC must be loaded from the A-Bus before instruction fetches can continue.

In addition to holding a slave PC, the memory controller contains a 3 byte queue to help speed execution. This queue is transparent to the RALU and to the user unless wait states are forced during external bus cycles. The instruction execution times shown in Section 14.8 show the normal execution times with no wait states added and the 16-bit bus selected. Reloading the slave PC and fetching the first byte of the new instruction stream takes 4 state times. This is reflected in the jump taken/not-taken times shown in the table.

2.7 System Bus

There are several operating modes on the 8096BH. The standard bus mode uses a 16-bit multiplexed address/data bus. Other bus modes include an 8-bit mode and a mode in which the bus size can dynamically be

switched between 8-bits and 16-bits. In addition, there are several options available on the type of control signals used by the bus. 8X9X devices only operate in the standard mode.

In the standard mode, external memory is addressed through lines AD0 through AD15 which form a 16-bit multiplexed (address/data) data bus. These lines share pins with I/O Ports 3 and 4. The falling edge of the Address Latch Enable (ALE) line is used to provide a clock to a transparent latch (74LS373) in order to demultiplex the bus. A typical circuit and the required timings are shown in Section 7 of the Hardware Design chapter. Since the 8096BH's external memory can be addressed as either bytes or words, the decoding is controlled with two lines, Bus High Enable (BHE) and Address/Data Line 0 (AD0). On the 8X9X devices the BHE line must be transparently latched, just as the addresses are.

To avoid confusion during the explanation of the memory system it is reasonable to give names to the demultiplexed address/data signals. The address signals will be called MA0 through MA15 (Memory Address), and the data signals will be called MD0 through MD15 (Memory Data).

When BHE is active (low), the memory connected to the high byte of the data bus should be selected. When MA0 is low the memory connected to the low byte of

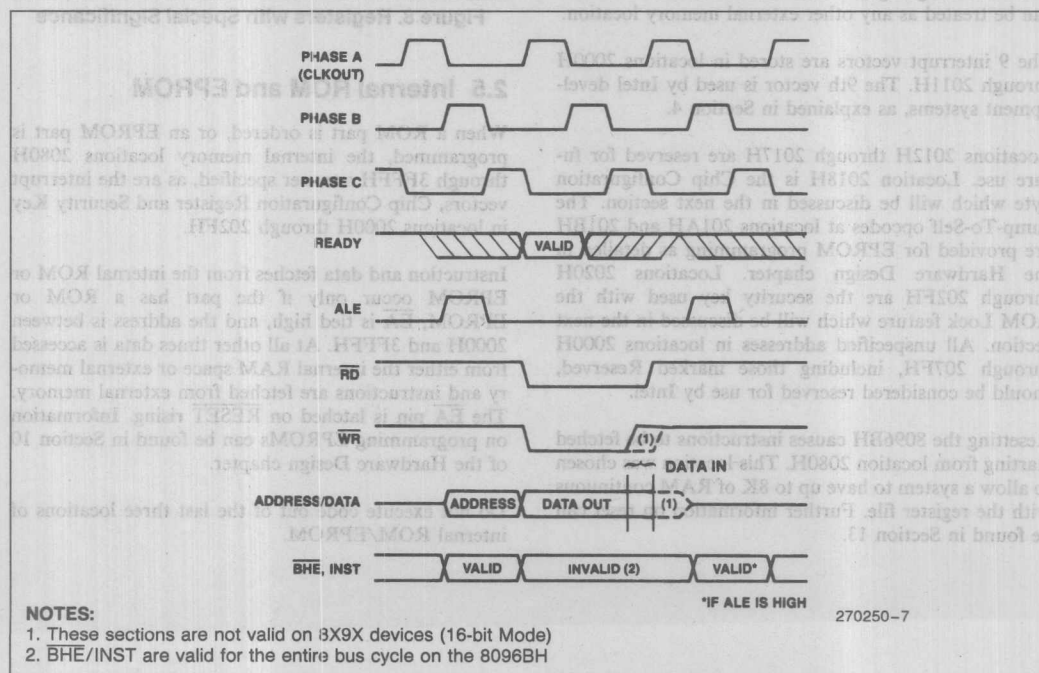


Figure 9. External Memory Timings

the data bus should be selected. In this way accesses to a 16-bit wide memory can be to the low (even) byte only ($\overline{\text{MA0}}=0$, $\overline{\text{BHE}}=1$), to the high (odd) byte only ($\overline{\text{MA0}}=1$, $\overline{\text{BHE}}=0$), or to both bytes ($\overline{\text{MA0}}=0$, $\overline{\text{BHE}}=0$). When a memory block is being used only for reads, $\overline{\text{BHE}}$ and $\overline{\text{MA0}}$ need not be decoded.

TIMINGS

Figure 9 shows the idealized waveforms related to the following description of external memory manipulations. For exact timing specifications please refer to the latest data sheet. When an external memory fetch begins, the address latch enable (ALE) line rises, the address is put on AD0-AD15 and $\overline{\text{BHE}}$ is set to the required state. ALE then falls, the address is taken off the pins, and the $\overline{\text{RD}}$ (Read) signal goes low. When $\overline{\text{RD}}$ falls, external memory should present its data to the 8096BH.

READ

The data from the external memory must be on the bus and stable for a minimum of the specified set-up time before the rising edge of $\overline{\text{RD}}$. The rising edge of $\overline{\text{RD}}$ latches the information into the 8096BH. If the read is for data, the INST pin will be low when the address is valid, if it is for an instruction the INST pin will be high during this time. The 48-lead part does not have the INST pin. The INST pin will be low for the Chip Configuration Byte and Interrupt Vector fetches.

WRITE

Writing to external memory requires timings that are similar to those required when reading from it. The main difference is that the write ($\overline{\text{WR}}$) signal is used instead of the $\overline{\text{RD}}$ signal. The timings are the same until the falling edge of the $\overline{\text{WR}}$ line. At this point the 8096BH removes the address and places the data on the bus. When the $\overline{\text{WR}}$ line goes high the data should be latched to the external memory. In systems which can write to byte locations, the AD0 and $\overline{\text{BHE}}$ lines must be used to decode $\overline{\text{WR}}$ into $\overline{\text{WR}}_{\text{Low byte}}$ ($\overline{\text{WRL}}$) and $\overline{\text{WR}}_{\text{High byte}}$ ($\overline{\text{WRH}}$) signals. INST is always low during a write, as instructions cannot be written. The exact timing specifications for memory accesses can be found in the data sheet.

READY

A ready line is available on the 8096BH to extend the width of the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ pulses in order to allow access of slow memories or for DMA purposes. If the READY line is low by the specified time after ALE falls, the 8096BH will hold the bus lines to their values at the falling edge of CLKOUT. When the READY line rises the bus cycle will continue with the next falling edge of CLKOUT.

Since the bus is synchronized to CLKOUT, it can be held only for an integral number of state times. If more than TLYH nanoseconds are added the processor will act unpredictably.

There are several set-up and hold times associated with the READY signal. If these timings are not met, the part may not respond with the proper number of wait states.

For falling edges of READY, sampling is done internally on the falling edge of Phase A. Since Phase A generates CLKOUT, (after some propagation delay) the sample will be taken prior to CLKOUT falling. The timing specification for this is given as TLLYV, the time between when ALE falls and READY must be valid. If READY changes between TLLYV max and the falling edge of CLKOUT (TLLYH MIN on 48-lead devices) it would be possible to have the READY signal transitioning as it is being sampled.

This situation could cause a metastable condition which could make the device operate unpredictably.

For the rising edge of READY, sampling is done internally on the rising edge of Phase A. The rising edge logic is fully synchronized, so it is not possible to cause a metastable condition once the device is in a valid not-ready condition. To cause one wait state to occur the rising edge of READY must occur before TLLYH MAX after ALE falls. If the signal is brought up after this time two wait states may occur. If two wait states are desired, READY should be brought high within the TLLYH specification + 3 TOSC. Additional wait states can be caused by adding additional state times to the READY low time. The maximum amount of time that a device may be held not-ready is specified as TYLYH.

The 8096BH has the ability to internally limit the number of wait states to 1, 2, or 3 as determined by the value in the Chip Configuration Register, (CCR). Using the CCR for ready timing is discussed at the end of this section. If a ready limit is set, the TLLYH MAX specification is not used. *The 8X9X devices do not have internal ready control.*

OPERATING MODES

The 8096BH supports a variety of options to simplify memory systems, interfacing requirements and ready control. Bus flexibility is provided by allowing selection of bus control signal definitions and runtime selection of the external bus width. In addition, several ready control modes are available to simplify the external hardware requirements for accessing slow devices. The Chip Configuration Register (CCR) is used to store the operating mode information.

Since there is no CCR on 8X9X devices, they can only be configured in the standard mode. This is the mode the 8096BH will run in if the CCR is loaded with 0FFH.

CHIP CONFIGURATION REGISTER (CCR)

Configuration information is stored in the Chip Configuration Register (CCR). Four of the bits in the register specify the bus control mode and ready control mode. Two bits also govern the level of ROM/EPROM protection and one bit is NANDed with the BUSWIDTH pin every bus cycle to determine the bus size. The CCR bit map is shown in Figure 10. The functions associated with each bit are described in this section.

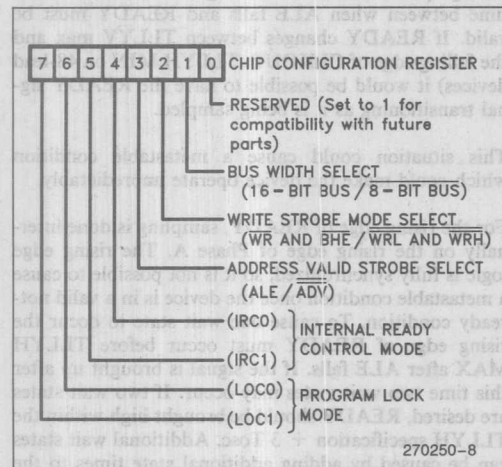


Figure 10. Chip Configuration Register

The CCR is loaded on reset with the Chip Configuration Byte, located at address 2018H. The CCR register is a non-memory mapped location that can only be written to during the reset sequence; once it is loaded it cannot be changed until the next reset occurs. The 8096BH will correctly read this location in every bus mode.

If the \overline{EA} pin is set to a logical 0, the access to 2018H comes from external memory. If \overline{EA} is a logical 1, the access comes from internal ROM/EPROM. If \overline{EA} is +12.5V, the CCR is loaded with a byte from a separate non-memory-mapped location called PCCB (Programming CCB). The Programming mode is described in Section 10 of the Hardware Design chapter.

The CCR is not present on 8X9X devices, but the Chip Configuration Byte at location 2018H should contain the hex value 0FFH to provide future compatibility. 8X9X devices do not access location 2018H on reset.

BUS WIDTH

The 8096BH external bus width can be run-time configured to operate as a standard 16-bit multiplexed address/data bus, or as an 8051 style 16-bit address/8-bit data bus.

During 16-bit bus cycles, Ports 3 and 4 contain the address multiplexed with data using ALE to latch the address. In 8-bit bus cycles, Port 3 is multiplexed address/data while Port 4 is address bits 8 through 15. The address bits on Port 4 are valid throughout an 8-bit bus cycle. Figure 11 shows the two options.

The bus width can be changed each bus cycle and is controlled using bit 1 of the CCR with the BUSWIDTH pin. If either CCR.1 or BUSWIDTH is a 0, external accesses will be over a 16-bit address/8-bit data bus. If both CCR.1 and BUSWIDTH are 1s, external accesses will be over a 16-bit address/16-bit data bus. Internal accesses are always 16-bits wide.

The bus width can be changed every external bus cycle if a 1 was loaded into CCR bit 1 at reset. If this is the case, changing the value of the BUSWIDTH pin at run-time will dynamically select the bus width. For example, the user could feed the INST line into the BUSWIDTH pin, thus causing instruction accesses to be word wide from EPROMs while data accesses are byte wide to and from RAMs. A second example would be to place an inverted version of Address bit 15 on the BUSWIDTH pin. This would make half of external memory word wide, while half is byte wide.

Since BUSWIDTH is sampled after address decoding has had time to occur, even more complex memory maps could be constructed. See the timing specifications for an exact description of BUSWIDTH timings. The bus width will be determined by bit 1 of the CCR alone on 48-pin parts since they do not have a BUSWIDTH pin.

When using an 8-bit bus, some performance degradation is to be expected. On the 8096BH, instruction execution times with an 8-bit bus will slow down if any of three conditions occur. First, word writes to external memory will cause the executing instruction to take two extra state times to complete. Second, word reads from external memory will cause a one state time extension of instruction execution time. Finally, if the pre-fetch queue is empty when an instruction fetch is requested, instruction execution is lengthened by one state time for each byte that must be externally acquired (worst case is the number of bytes in the instruction minus one.)

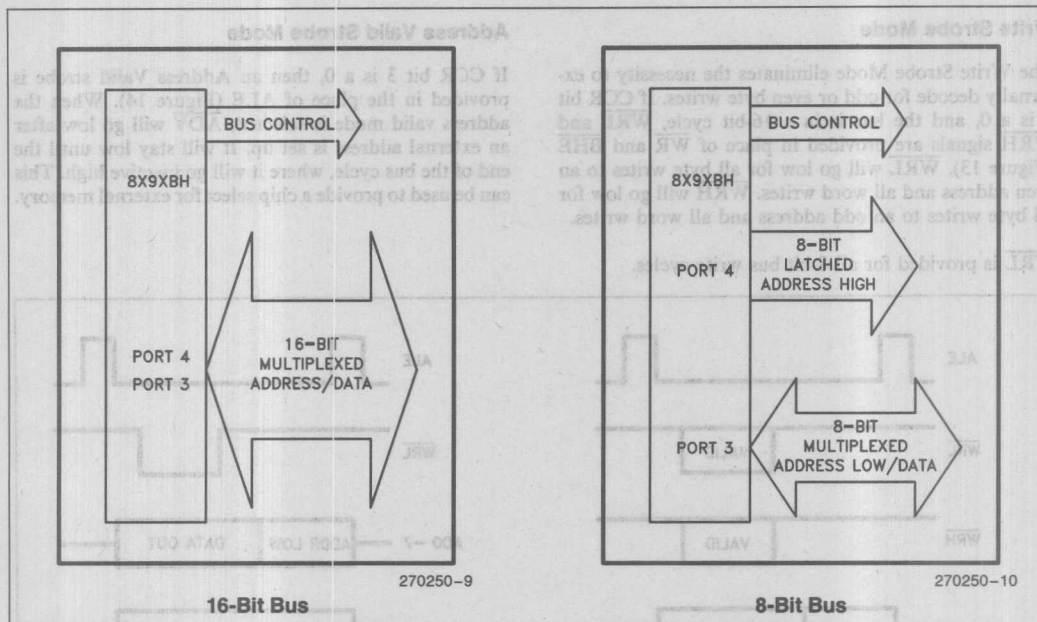


Figure 11. Bus Width Options

BUS CONTROL

Using the CCR, the 8096BH can be made to provide bus control signals of several types. Three control lines have dual functions designed to reduce external hardware. Bits 2 and 3 of the CCR specify the functions performed by these control lines. Figures 12-15 show the signals which can be modified by changing bits in the CCR, all other lines will operate as shown in Figure 9.

Standard Bus Control

If CCR bits 2 and 3 are 1s, then the standard 8096BH control signals \overline{WR} , \overline{BHE} and ALE are provided (Figure 12). \overline{WR} will come out for every write. \overline{BHE} will be valid throughout the bus cycle and can be combined with \overline{WR} and address line 0 to form \overline{WRL} and \overline{WRH} . ALE will rise as the address starts to come out, and will fall to provide the signal to externally latch the address. The control signals on 8X9X devices are similar, but not identical to those shown here. Figure 9 shows the 8X9X timings.

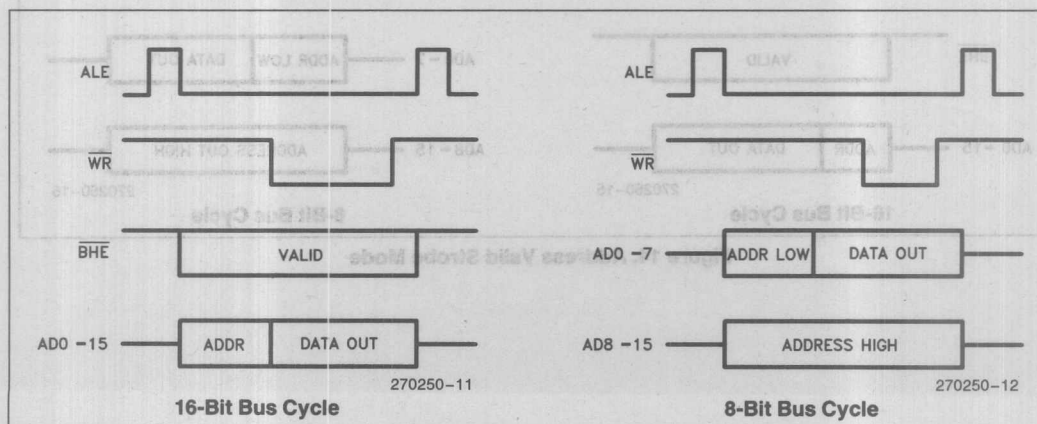


Figure 12. Standard Bus Control

Write Strobe Mode

The Write Strobe Mode eliminates the necessity to externally decode for odd or even byte writes. If CCR bit 2 is a 0, and the bus is in a 16-bit cycle, \overline{WRL} and \overline{WRH} signals are provided in place of \overline{WR} and \overline{BHE} (Figure 13). \overline{WRL} will go low for all byte writes to an even address and all word writes. \overline{WRH} will go low for all byte writes to an odd address and all word writes.

\overline{WRL} is provided for all 8-bit bus write cycles.

Address Valid Strobe Mode

If CCR bit 3 is a 0, then an Address Valid strobe is provided in the place of ALE (Figure 14). When the address valid mode is selected, \overline{ADV} will go low after an external address is set up. It will stay low until the end of the bus cycle, where it will go inactive high. This can be used to provide a chip select for external memory.

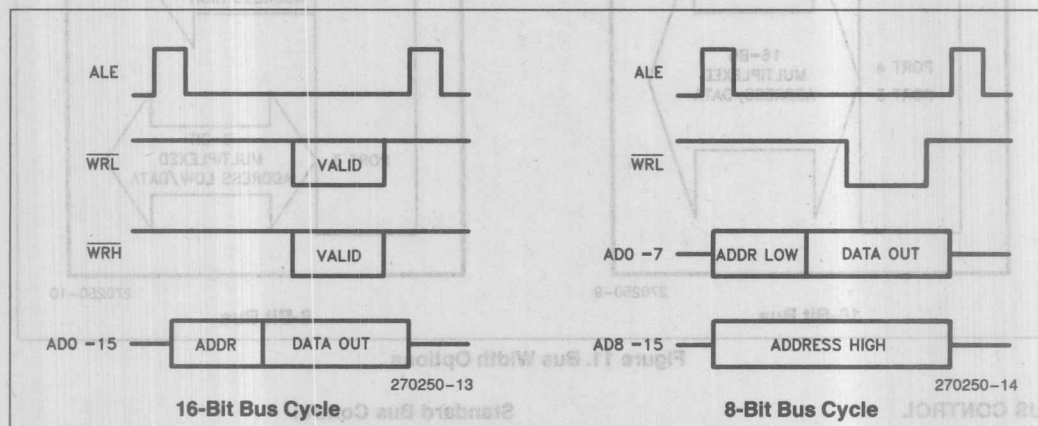


Figure 13. Write Strobe Mode

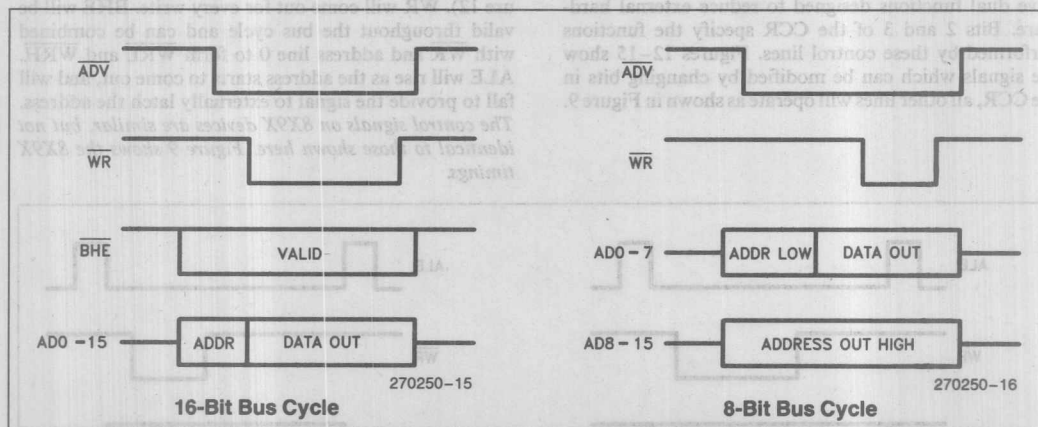


Figure 14. Address Valid Strobe Mode

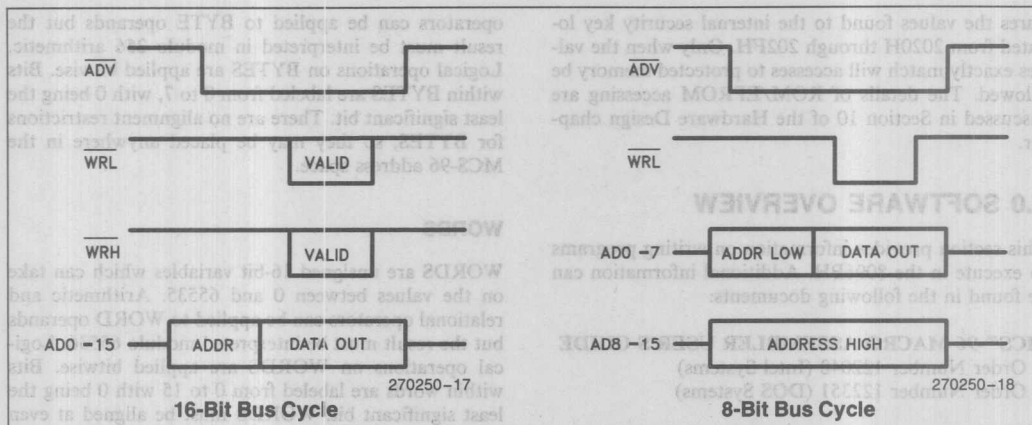


Figure 15. Write Strobe with Address Valid Strobe

Address Valid with Write Strobe

If both CCR bits 2 and 3 are 0s, both the Address Valid strobe and the Write Strobes will be provided for bus control. Figure 15 shows these signals.

READY CONTROL

To simplify ready control, four modes of internal ready control logic have been provided. The modes are chosen by properly configuring bits 4 and 5 of the CCR.

The internal ready control logic can be used to limit the number of wait states that slow devices can insert into the bus cycle. When the READY pin is pulled low, wait states will be inserted into the bus cycle until the READY pin goes high, or the number of wait states equals the number specified by CCR bits 4 and 5, whichever comes first. Table 1 shows the number of wait states that can be selected. Internal Ready control can be disabled by loading 11 into bits 4 and 5 of the CCR.

Table 1. Internal Ready Control

IRC1	IRC0	Description
0	0	Limit to 1 Wait State
0	1	Limit to 2 Wait States
1	0	Limit to 3 Wait States
1	1	Disable Internal Ready Control

This feature provides for simple ready control. For example, every slow memory chip select line could be ORed together and be connected to the READY pin with CCR bits 4 and 5 programmed to give the desired number of wait states to the slow devices.

ROM/EPROM LOCK

Four modes of program memory lock are available on the 839XBH and 879XBH parts. CCR bits 6 and 7 (LOC0, LOC1) select whether internal program memory can be read (or written in EPROM parts) by a program executing from external memory. The modes are shown in Table 2. Internal ROM/EPROM addresses 2020H through 3FFFH are protected from reads while 2000H through 3FFFH are protected from writes, as set by the CCR.

Table 2. Program Lock Modes

LOC1	LOC0	Protection
0	0	Read and Write Protected
0	1	Read Protected
1	0	Write Protected
1	1	No Protection

Only code executing from internal memory can read protected internal memory, while a write protected memory can not be written to, even from internal execution. As a result of 8096BH prefetching of instructions, however, accesses to protected memory are not allowed for instructions located above 3FFAH. This is because the lock protection mechanism is gated off of the Memory Controller's slave program counter and not the CPU program counter. If the bus controller receives a request to perform a read of protected memory, the read sequence occurs with indeterminate data being returned to the CPU. Note that the interrupt vectors and the CCR are not protected.

To provide verification and testing when the program lock feature is enabled, the 839XBH and 879XBH verify the security key before programming or test modes are allowed to read from protected memory. Before protected memory can be read, the chip reads external memory locations 4020H through 402FH and com-

compares the values found to the internal security key located from 2020H through 202FH. Only when the values exactly match will accesses to protected memory be allowed. The details of ROM/EPROM accessing are discussed in Section 10 of the Hardware Design chapter.

3.0 SOFTWARE OVERVIEW

This section provides information on writing programs to execute in the 8096BH. Additional information can be found in the following documents:

MCS®-96 MACRO ASSEMBLER USER'S GUIDE

Order Number 122048 (Intel Systems)
Order Number 122351 (DOS Systems)

MCS®-96 UTILITIES USER'S GUIDE

Order Number 122049 (Intel Systems)
Order Number 122356 (DOS Systems)

PL/M-96 USER'S GUIDE

Order Number 122134 (Intel Systems)
Order Number 122361 (DOS Systems)

Throughout this section, short sections of code are used to illustrate the operation of the device. For these sections it has been assumed that a set of temporary registers have been predeclared. The names of these registers have been chosen as follows:

AX, BX, CX, and DX are 16-bit registers.

AL is the low byte of AX, AH is the high byte.

BL is the low byte of BX

CL is the low byte of CX

DL is the low byte of DX

These are the same as the names for the general data registers used in the 8086BH. It is important to note, however, that in the 8096, these are not dedicated registers but merely the symbolic names assigned by the programmer to an eight byte region within the onboard register file.

3.1 Operand Types

The MCS®-96 architecture provides support for a variety of data types which are likely to be useful in a control application. In the discussion of these operand types that follows, the names adopted by the PLM-96 programming language will be used where appropriate. To avoid confusion, the name of an operand type will be capitalized. A "BYTE" is an unsigned eight bit variable; a "byte" is an eight bit unit of data of any type.

BYTES

BYTES are unsigned 8-bit variables which can take on the values between 0 and 255. Arithmetic and relational

operators can be applied to BYTE operands but the result must be interpreted in modulo 256 arithmetic. Logical operations on BYTES are applied bitwise. Bits within BYTES are labeled from 0 to 7, with 0 being the least significant bit. There are no alignment restrictions for BYTES, so they may be placed anywhere in the MCS-96 address space.

WORDS

WORDS are unsigned 16-bit variables which can take on the values between 0 and 65535. Arithmetic and relational operators can be applied to WORD operands but the result must be interpreted modulo 65536. Logical operations on WORDS are applied bitwise. Bits within words are labeled from 0 to 15 with 0 being the least significant bit. WORDS must be aligned at even byte boundaries in the MCS-96 address space. The least significant byte of the WORD is in the even byte address and the most significant byte is in the next higher (odd) address. The address of a word is the address of its least significant byte. Word operations to odd addresses are not guaranteed to operate in a consistent manner.

SHORT-INTEGERS

SHORT-INTEGERS are 8-bit signed variables which can take on the values between -128 and +127. Arithmetic operations which generate results outside of the range of a SHORT-INTEGER will set the overflow indicators in the program status word. The actual numeric result returned will be the same as the equivalent operation on BYTE variables. There are no alignment restrictions on SHORT-INTEGERS so they may be placed anywhere in the MCS-96 address space.

INTEGERS

INTEGERS are 16-bit signed variables which can take on the values between -32,768 and 32,767. Arithmetic operations which generate results outside of the range of an INTEGER will set the overflow indicators in the program status word. The actual numeric result returned will be the same as the equivalent operation on WORD variables. INTEGERS conform to the same alignment and addressing rules as do WORDS.

BITS

BITS are single-bit operands which can take on the Boolean values of true and false. In addition to the normal support for bits as components of BYTE and WORD operands, the 8096 provides for the direct testing of any bit in the internal register file. The MCS-96 architecture requires that bits be addressed as components of BYTES or WORDS, it does not support the direct addressing of bits that can occur in the MCS-51 architecture.

DOUBLE-WORDS

DOUBLE-WORDS are unsigned 32-bit variables which can take on the values between 0 and 4,294,967,295. The MCS-96 architecture provides direct support for this operand type only for shifts and as the dividend in a 32 by 16 divide and the product of a 16 by 16 multiply. For these operations a DOUBLE-WORD variable must reside in the on-board register file of the 8096 and be aligned at an address which is evenly divisible by 4. A DOUBLE-WORD operand is addressed by the address of its least significant byte. DOUBLE-WORD operations which are not directly supported can be easily implemented with two WORD operations. For consistency with Intel provided software the user should adopt the conventions for addressing DOUBLE-WORD operands which are discussed in Section 3.5.

3.2 Operand Addressing

Operands are accessed within the address space of the 8096 with one of six basic addressing modes. Some of the details of how these addressing modes work are hidden by the assembly language. If the programmer is to take full advantage of the architecture, it is important that these details be understood. This section will describe the addressing modes as they are handled by the hardware. At the end of this section the addressing

REGISTER-DIRECT REFERENCES

The register-direct mode is used to directly access a register from the 256 byte on-board register file. The register is selected by an 8-bit field within the instruction and register address and must conform to the

Examples

ADD	AX, BX, CX	; AX := BX + CX
MUL	AX, BX	; AX := AX * BX
INCB	CL	; CL := CL + 1

INDIRECT REFERENCES

The indirect mode is used to access an operand by placing its address in a WORD variable in the register file. The calculated address must conform to the alignment rules for the operand type. Note that the indirect address can refer to an operand anywhere within the address space of the 8096, including the register file. The

LONG-INTEGERS

LONG-INTEGERS are 32-bit signed variables which can take on the values between -2,147,483,648 and 2,147,483,647. The MCS-96 architecture provides direct support for this data type only for shifts and as the dividend in a 32 by 16 divide and the product of a 16 by 16 multiply.

LONG-INTEGERS can also be normalized. For these operations a LONG-INTEGER variable must reside in the onboard register file of the 8096 and be aligned at an address which is evenly divisible by 4. A LONG-INTEGER is addressed by the address of its least significant byte.

LONG-INTEGER operations which are not directly supported can be easily implemented with two INTEGER operations. For consistency with Intel provided software, the user should adopt the conventions for addressing LONG operands which are discussed in Section 3.5.

modes will be described as they are seen through the assembly language. The six basic address modes which will be described are termed register-direct, indirect, indirect with auto-increment, immediate, short-indexed, and long-indexed. Several other useful addressing operations can be achieved by combining these basic addressing modes with specific registers such as the ZERO register or the stack pointer.

alignment rules for the operand type. Depending on the instruction, up to three registers can take part in the calculation.

Examples

LD	AX, [AX]	; AX := MEM_WORD (AX)
ADDB	AL, BL, [CX]	; AL := BL + MEM_BYTE (CX)
POP	[AX]	; MEM_WORD (AX) := MEM_WORD (SP) ; SP := SP + 2

INDIRECT WITH AUTO-INCREMENT REFERENCES

This addressing mode is the same as the indirect mode except that the WORD variable which contains the indirect address is incremented *after* it is used to address the operand. If the instruction operates on BYTES or

SHORT-INTEGERS the indirect address variable will be incremented by one, if the instruction operates on WORDS or INTEGERS the indirect address variable will be incremented by two.

Examples

```
LD    AX, [BX]+      ; AX:=MEM_WORD(BX) ; BX:=BX+2
ADDB  AL, BL, [CX]+   ; AL:=BL+MEM_BYTE(CX) ; CX:=CX+1
PUSH  [AX]+          ; SP:=SP-2;
                        ; MEM_WORD(SP) :=MEM_WORD(AX)
                        ; AX:=AX+2
```

IMMEDIATE REFERENCES

This addressing mode allows an operand to be taken directly from a field in the instruction. For operations on BYTE or SHORT-INTEGERS this field is eight bits wide, for operations on WORD or INTE-

GER operands the field is 16 bits wide. An instruction can contain only one immediate reference and the remaining operand(s) must be register-direct references.

Examples

```
ADD  AX, #340 ; AX:=AX+340
PUSH #1234H ; SP:=SP-2; MEM_WORD(SP) :=1234H
DIVB AX, #10 ; AL:=AX/10 ; AH:=AX MOD 10
```

SHORT-INDEXED REFERENCES

In this addressing mode an eight bit field in the instruction selects a WORD variable in the register file which is assumed to contain an address. A second eight bit field in the instruction stream is sign-extended and summed with the WORD variable to form the address of the operand which will take part in the calculation.

Since the eight bit field is sign-extended, the effective address can be up to 128 bytes before the address in the WORD variable and up to 127 bytes after it. An instruction can contain only one short-indexed reference and the remaining operand(s) must be register-direct references.

Examples

```
LD    AX, 12[BX] ; AX:=MEM_WORD(BX+12)
MULB  AX, BL, 3[CX] ; AX:=BL*MEM_BYTE(CX+3)
```

LONG-INDEXED REFERENCES

This addressing mode is like the short-indexed mode except that a 16-bit field is taken from the instruction and added to the WORD variable to form the address of the operand. No sign extension is necessary. An in-

struction can contain only one long-indexed reference and the remaining operand(s) must be register-direct references.

Examples

```
AND  AX, BX, TABLE[CX] ; AX:=BX AND MEM_WORD(TABLE+CX)
ST   AX, TABLE[BX] ; MEM_WORD(TABLE+BX) :=AX
ADDB AL, BL, LOOKUP[CX] ; AL:=BL+MEM_BYTE(LOOKUP+CX)
```

ZERO REGISTER ADDRESSING

The first two bytes in the register file are fixed at zero by the 8096 hardware. In addition to providing a fixed source of the constant zero for calculations and comparisons, this register can be used as the WORD variable in a long-indexed reference.

Examples

```
ADD AX,1234[0] ; AX:=AX+MEM_WORD(1234)
POP 5678[0] ; MEM_WORD(5678):=MEM_WORD(SP)
; SP:=SP+2
```

STACK POINTER REGISTER ADDRESSING

The system stack pointer in the 8096 can be accessed as register 18H of the internal register file. In addition to providing for convenient manipulation of the stack pointer, this also facilitates the accessing of operands in the stack. The top of the stack, for example, can be

Examples

```
PUSH [SP] ; DUPLICATE TOP_OF_STACK
LD AX,2[SP] ; AX:=NEXT_TO_TOP
```

ASSEMBLY LANGUAGE ADDRESSING MODES

The 8096 assembly language simplifies the choice of addressing modes to be used in several respects:

Direct Addressing. The assembly language will choose between register-direct addressing and long-indexed with the ZERO register depending on where the operand is in memory. The user can simply refer to an operand by its symbolic name; if the operand is in the register file, a register-direct reference will be used, if the operand is elsewhere in memory, a long-indexed reference will be generated.

Indexed Addressing. The assembly language will choose between short and long indexing depending on the value of the index expression. If the value can be expressed in eight bits then short indexing will be used, if it cannot be expressed in eight bits then long indexing will be used.

able in a long-indexed reference. This combination of register selection and address mode allows any location in memory to be addressed directly.

accessed by using the stack pointer as the WORD variable in an indirect reference. In a similar fashion, the stack pointer can be used in the short-indexed mode to access data within the stack.

The use of these features of the assembly language simplifies the programming task and should be used whenever possible.

3.3 Program Status Word

The program status word (PSW) is a collection of Boolean flags which retain information concerning the state of the user's program. The format of the PSW is shown in Figure 16. The information in the PSW can be broken down into two basic categories; interrupt control and condition flags. The PSW can be saved in the system stack with a single operation (PUSHF) and restored in a like manner (POPF).

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Z	N	V	VT	C	—	I	ST	<Interrupt Mask Reg>							

Figure 16. PSW Register

INTERRUPT FLAGS

The lower eight bits of the PSW are used to individually mask the various sources of interrupt to the 8096. A logical '1' in these bit positions enables the servicing of the corresponding interrupt. These mask bits can be accessed as an eight bit byte (INT_MASK—address 8) in the on-board register file. Bit 9 in the PSW is the global interrupt disable. If this bit is cleared then all interrupts will be locked out except for the Non Maskable Interrupt (NMI). Note that the various interrupts are collected in the INT_PENDING register even if they are locked out. Execution of the corresponding service routines will proceed according to their priority when they become enabled. Further information on the interrupt structure of the 8096 can be found in Section 4.

CONDITION FLAGS

The remaining bits in the PSW are set as side effects of instruction execution and can be tested by the conditional jump instructions.

Z. The Z (Zero) flag is set to indicate that the operation generated a result equal to zero. For the add-with-carry (ADDC) and subtract-with-borrow (SUBC) operations the Z flag is cleared if the result is non-zero but is never set. These two instructions are normally used in conjunction with the ADD and SUB instructions to perform multiple precision arithmetic. The operation of the Z flag for these instructions leaves it indicating the proper result for the entire multiple precision calculation.

N. The N (Negative) flag is set to indicate that the operation generated a negative result. Note that the N flag will be set to the algebraically correct state even if the calculation overflows. When the NEGB instruction is performed on a byte register containing 80H, or the NEG instruction is performed on a word register containing 8000H, the N flag is set.

V. The V (overflow) flag is set to indicate that the operation generated a result which is outside the range that can be expressed in the destination data type. For the SHL, SHLB and SHLL instructions, the V flag will be set if the most significant bit of the operand changes at any time during the shift.

VT. The VT (oVerflow Trap) flag is set whenever the V flag is set but can only be cleared by an instruction which explicitly operates on it such as the CLRVT or JVT instructions. The operation of the VT flag allows for the testing for a possible overflow condition at the end of a sequence of related arithmetic operations. This

is normally more efficient than testing the V flag after each instruction.

C. The C (Carry) flag is set to indicate the state of the arithmetic carry from the most significant bit of the ALU for an arithmetic operation or the state of the last bit shifted out of the operand for a shift. Arithmetic Borrow after a subtract operation is the complement of the C flag (i.e. if the operation generated a borrow then $C = 0$).

ST. The ST (STicky bit) flag is set to indicate that during a right shift a 1 has been shifted first into the C flag and then been shifted out. The ST flag is undefined after a multiply operation. The ST flag can be used along with the C flag to control rounding after a right shift. Consider multiplying two eight bit quantities and then scaling the result down to 12 bits:

```
MULUB  AX,CL,DL  ;AX:=CL*DL
SHR    AX,#4      ;Shift right 4 places
```

If the C flag is set after the shift, it indicates that the bits shifted off the end of the operand were greater-than or equal-to one half the least significant bit (LSB) of the result. If the C flag is clear after the shift, it indicates that the bits shifted off the end of the operand were less than half the LSB of the result. Without the ST flag, the rounding decision must be made on the basis of this information alone. (Normally the result would be rounded up if the C flag is set.) The ST flag allows a finer resolution in the rounding decision:

C ST	Value of the Bits Shifted Off
0 0	Value = 0
0 1	$0 < \text{Value} < \frac{1}{2} \text{LSB}$
1 0	Value = $\frac{1}{2} \text{LSB}$
1 1	Value $> \frac{1}{2} \text{LSB}$

Figure 17. Rounding Alternatives

Imprecise rounding can be a major source of error in a numerical calculation; use of the ST flag improves the options available to the programmer.

3.4 Instruction Set

The MCS-96 instruction set contains a full set of arithmetic and logical operations for the 8-bit data types BYTE and SHORT INTEGER and for the 16-bit data types WORD and INTEGER. The DOUBLE-WORD and LONG data types (32 bits) are supported for the products of 16 by 16 multiplies and the dividends of 32

Table 3. Instruction Summary

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	$D \leftarrow D + A$	✓	✓	✓	✓	↑	—	
ADD/ADDB	3	$D \leftarrow B + A$	✓	✓	✓	✓	↑	—	
ADDC/ADDCB	2	$D \leftarrow D + A + C$	↓	✓	✓	✓	↑	—	
SUB/SUBB	2	$D \leftarrow D - A$	✓	✓	✓	✓	↑	—	
SUB/SUBB	3	$D \leftarrow B - A$	✓	✓	✓	✓	↑	—	
SUBC/SUBCB	2	$D \leftarrow D - A + C - 1$	↓	✓	✓	✓	↑	—	
CMP/CMPB	2	$D - A$	✓	✓	✓	✓	↑	—	
MUL/MULU	2	$D, D + 2 \leftarrow D * A$	—	—	—	—	—	?	2
MUL/MULU	3	$D, D + 2 \leftarrow B * A$	—	—	—	—	—	?	2
MULB/MULUB	2	$D, D + 1 \leftarrow D * A$	—	—	—	—	—	?	3
MULB/MULUB	3	$D, D + 1 \leftarrow B * A$	—	—	—	—	—	?	3
DIVU	2	$D \leftarrow (D, D + 2)/A, D + 2 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	2
DIVUB	2	$D \leftarrow (D, D + 1)/A, D + 1 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	3
DIV	2	$D \leftarrow (D, D + 2)/A, D + 2 \leftarrow \text{remainder}$	—	—	—	?	↑	—	
DIVB	2	$D \leftarrow (D, D + 1)/A, D + 1 \leftarrow \text{remainder}$	—	—	—	?	↑	—	
AND/ANDB	2	$D \leftarrow D \text{ and } A$	✓	✓	0	0	—	—	
AND/ANDB	3	$D \leftarrow B \text{ and } A$	✓	✓	0	0	—	—	
OR/ORB	2	$D \leftarrow D \text{ or } A$	✓	✓	0	0	—	—	
XOR/XORB	2	$D \leftarrow D \text{ (excl. or) } A$	✓	✓	0	0	—	—	
LD/LDB	2	$D \leftarrow A$	—	—	—	—	—	—	
ST/STB	2	$A \leftarrow D$	—	—	—	—	—	—	
LDBSE	2	$D \leftarrow A; D + 1 \leftarrow \text{SIGN}(A)$	—	—	—	—	—	—	3, 4
LDBZE	2	$D \leftarrow A; D + 1 \leftarrow 0$	—	—	—	—	—	—	3, 4
PUSH	1	$SP \leftarrow SP - 2; (SP) \leftarrow A$	—	—	—	—	—	—	
POP	1	$A \leftarrow (SP); SP \leftarrow SP + 2$	—	—	—	—	—	—	
PUSHF	0	$SP \leftarrow SP - 2; (SP) \leftarrow PSW;$ $PSW \leftarrow 0000H$ $I \leftarrow 0$	0	0	0	0	0	0	
POPF	0	$PSW \leftarrow (SP); SP \leftarrow SP + 2;$ $I \leftarrow \text{✓}$	✓	✓	✓	✓	✓	✓	
SJMP	1	$PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LJMP	1	$PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5
BR [indirect]	1	$PC \leftarrow (A)$	—	—	—	—	—	—	
SCALL	1	$SP \leftarrow SP - 2; (SP) \leftarrow PC;$ $PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LCALL	1	$SP \leftarrow SP - 2; (SP) \leftarrow PC;$ $PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5
RET	0	$PC \leftarrow (SP); SP \leftarrow SP + 2$	—	—	—	—	—	—	
J (conditional)	1	$PC \leftarrow PC + 8\text{-bit offset (if taken)}$	—	—	—	—	—	—	5
JC	1	Jump if C = 1	—	—	—	—	—	—	5
JNC	1	Jump if C = 0	—	—	—	—	—	—	5
JE	1	Jump if Z = 1	—	—	—	—	—	—	5

NOTES:

1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
2. D, D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D, D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to a word.
5. Offset is a 2's complement number.

Table 3. Instruction Summary (Continued)

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
JNE	1	Jump if Z = 0	—	—	—	—	—	—	5
JGE	1	Jump if N = 0	—	—	—	—	—	—	5
JLT	1	Jump if N = 1	—	—	—	—	—	—	5
JGT	1	Jump if N = 0 and Z = 0	—	—	—	—	—	—	5
JLE	1	Jump if N = 1 or Z = 1	—	—	—	—	—	—	5
JH	1	Jump if C = 1 and Z = 0	—	—	—	—	—	—	5
JNH	1	Jump if C = 0 or Z = 1	—	—	—	—	—	—	5
JV	1	Jump if V = 1	—	—	—	—	—	—	5
JNV	1	Jump if V = 0	—	—	—	—	—	—	5
JVT	1	Jump if VT = 1; Clear VT	—	—	—	—	0	—	5
JNVT	1	Jump if VT = 0; Clear VT	—	—	—	—	0	—	5
JST	1	Jump if ST = 1	—	—	—	—	—	—	5
JNST	1	Jump if ST = 0	—	—	—	—	—	—	5
JBS	3	Jump if Specified Bit = 1	—	—	—	—	—	—	5, 6
JBC	3	Jump if Specified Bit = 0	—	—	—	—	—	—	5, 6
DJNZ	1	D ← D - 1; if D ≠ 0 then PC ← PC + 8-bit offset	—	—	—	—	—	—	5
DEC/DECB	1	D ← D - 1	✓	✓	✓	✓	↑	—	
NEG/NEGB	1	D ← 0 - D	✓	✓	✓	✓	↑	—	
INC/INCB	1	D ← D + 1	✓	✓	✓	✓	↑	—	
EXT	1	D ← D; D + 2 ← Sign(D)	✓	✓	0	0	—	—	2
EXTB	1	D ← D; D + 1 ← Sign(D)	✓	✓	0	0	—	—	3
NOT/NOTB	1	D ← Logical Not(D)	✓	✓	0	0	—	—	
CLR/CLRB	1	D ← 0	1	0	0	0	—	—	
SHL/SHLB/SHLL	2	C ← msb ———— lsb ← 0	✓	?	✓	✓	↑	—	7
SHR/SHRB/SHRL	2	0 → msb ———— lsb → C	✓	?	✓	0	—	✓	7
SHRA/SHRAB/SHRAL	2	msb → msb ———— lsb → C	✓	✓	✓	0	—	✓	7
SETC	0	C ← 1	—	—	1	—	—	—	
CLRC	0	C ← 0	—	—	0	—	—	—	
CLRVT	0	VT ← 0	—	—	—	—	0	—	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	—	—	—	—	—	—	
EI	0	Enable All Interrupts (I ← 1)	—	—	—	—	—	—	
NOP	0	PC ← PC + 1	—	—	—	—	—	—	
SKIP	0	PC ← PC + 2	—	—	—	—	—	—	
NORML	2	Left shift till msb = 1; D ← shift count	✓	?	0	—	—	—	7
TRAP	0	SP ← SP - 2; (SP) ← PC PC ← (2010H)	—	—	—	—	—	—	9

NOTES:

1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
2. Offset is a 2's complement number.
3. Specified bit is one of the 2048 bits in the register file.
4. The "L" (Long) suffix indicates double-word operation.
5. Initiates a Reset by pulling RESET low. Software should re-initialize all the necessary registers with code starting at 2080H.
6. The assembler will not accept this mnemonic.

3.5 Software Standards and Conventions

For a software project of any size it is a good idea to modularize the program and to establish standards which control the communication between these modules. The nature of these standards will vary with the needs of the final application. A common component of all of these standards, however, must be the mechanism for passing parameters to procedures and returning results from procedures. In the absence of some overriding consideration which prevents their use, it is suggested that the user conform to the conventions adopted by the PLM-96 programming language for procedure linkage. It is a very usable standard for both the assembly language and PLM-96 environment and it offers compatibility between these environments. Another advantage is that it allows the user access to the same floating point arithmetics library that PLM-96 uses to operate on REAL variables.

REGISTER UTILIZATION

The MCS-96 architecture provides a 256 byte register file. Some of these registers are used to control register-mapped I/O devices and for other special functions such as the ZERO register and the stack pointer. The remaining bytes in the register file, some 230 of them, are available for allocation by the programmer. If these registers are to be used effectively, some overall strategy for their allocation must be adopted. PLM-96 adopts the simple and effective strategy of allocating the eight bytes between addresses 1CH and 23H as temporary storage. The starting address of this region is called PLMREG. The remaining area in the register file is treated as a segment of memory which is allocated as required.

ADDRESSING 32-BIT OPERANDS

These operands are formed from two adjacent 16-bit words in memory. The least significant word of the double word is always in lower address, even when the data is in the stack (which means that the most significant word must be pushed into the stack first). A double word is addressed by the address of its least significant byte. Note that the hardware supports some operations on double words (e.g. normalize and divide). For these operations the double word must be in the internal register file and must have an address which is evenly divisible by four.

SUBROUTINE LINKAGE

Parameters are passed to subroutines in the stack. Parameters are pushed into the stack in the order that they are encountered in the scanning of the source text. Eight-bit parameters (BYTES or SHORT-INTegers) are pushed into the stack with the high order

byte undefined. Thirty-two bit parameters (LONG-INTegers, DOUBLE-WORDS, and REALS) are pushed into the stack as two 16-bit values; the most significant half of the parameter is pushed into the stack first.

As an example, consider the following PLM-96 procedure:

```
example_procedure: PROCEDURE
(param1,param2,param3);
  DECLARE param1 BYTE,
           param2 DWORD,
           param3 WORD;
```

When this procedure is entered at run time the stack will contain the parameters in the following order:

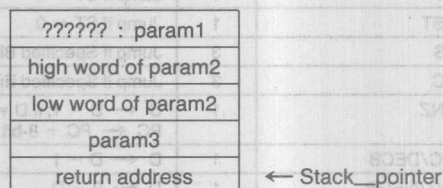


Figure 18. Stack Image

If a procedure returns a value to the calling code (as opposed to modifying more global variables) then the result is returned in the variable PLMREG. PLMREG is viewed as either an 8-, 16- or 32-bit variable depending on the type of the procedure.

The standard calling convention adopted by PLM-96 has several key features:

- Procedures can always assume that the eight bytes of register file memory starting at PLMREG can be used as temporaries within the body of the procedure.
- Code which calls a procedure must assume that the eight bytes of register file memory starting at PLMREG are modified by the procedure.
- The Program Status Word (PSW—see Section 3.3) is not saved and restored by procedures so the calling code must assume that the condition flags (Z, N, V, VT, C, and ST) are modified by the procedure.
- Function results from procedures are always returned in the variable PLMREG.

PLM-96 allows the definition of INTERRUPT procedures which are executed when a predefined interrupt occurs. These procedures do not conform to the rules of a normal procedure. Parameters cannot be passed to these procedures and they cannot return results. Since they can execute essentially at any time (hence the term interrupt), these procedures must save the PSW and PLMREG when they are entered and restore these values before they exit.

4.0 INTERRUPT STRUCTURE

There are 21¹ sources of interrupts on the 8096BH. These sources are gathered into 8 interrupt types as indicated in Figure 19. The I/O control registers which control some of the sources are indicated in the figure. Each of the eight types of interrupts has its own interrupt vector as listed in Figure 20. In addition to the 8 standard interrupts, there is a TRAP instruction which acts as a software generated interrupt. This instruction is not currently supported by the MCS-96 Assembler and is reserved for use in Intel development systems.

The programmer must initialize the interrupt vector table with the starting address of the appropriate interrupt service routine. It is suggested that any unused interrupts be vectored to an error handling routine. The error routine should contain recovery code that will not further corrupt an already erroneous situation. In a debug environment, it may be desirable to have the routine lock into a jump to self loop which would be easily traceable with emulation tools. More sophisticated routines may be appropriate for production code recoveries.

Three registers control the operation of the interrupt system: Interrupt Pending, Interrupt Mask, and the

PSW which contains a global disable bit. A block diagram of the system is shown in Figure 21. The transition detector looks for 0 to 1 transitions on any of the sources. External sources have a maximum transition speed of one edge every state time. If this is exceeded the interrupt may not be detected.

Vector	Vector Location		Priority
	(High Byte)	(Low Byte)	
Software	2011H	2010H	Not Applicable
Extint	200FH	200EH	7 (Highest)
Serial Port	200DH	200CH	6
Software	200BH	200AH	5
Timers			
HSI.0	2009H	2008H	4
High Speed	2007H	2006H	3
Outputs			
HSI Data	2005H	2004H	2
Available			
A/D Conversion	2003H	2002H	1
Complete			
Timer Overflow	2001H	2000H	0 (Lowest)

Figure 20. Interrupt Vector Locations

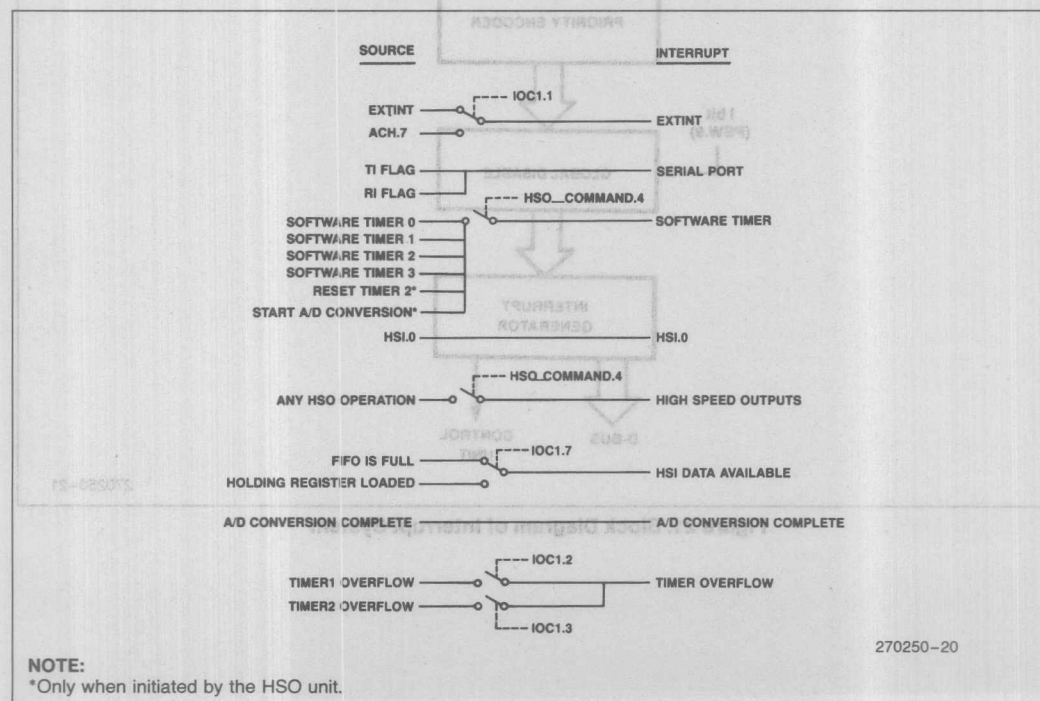


Figure 19. All Possible Interrupt Sources

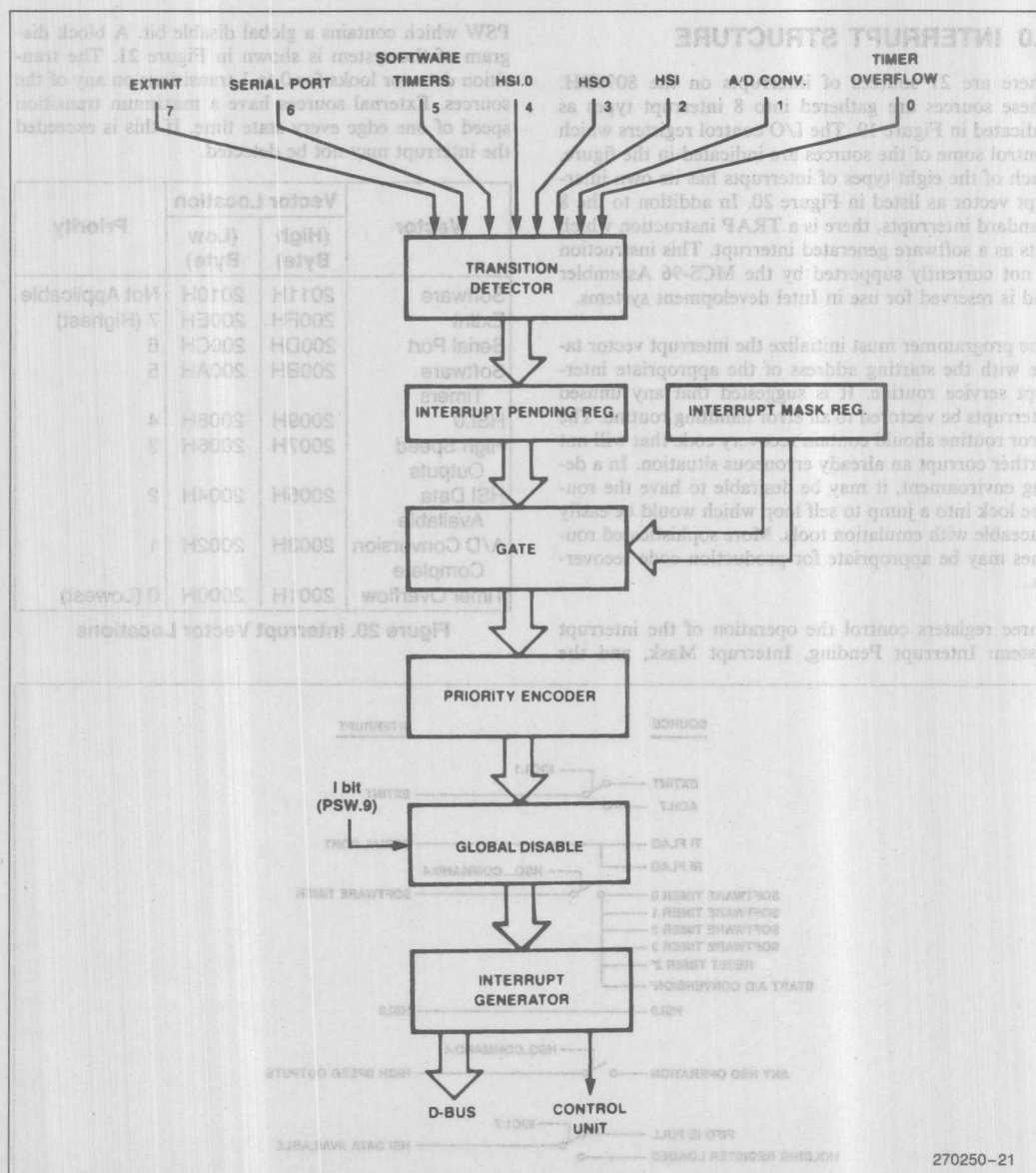


Figure 21. Block Diagram of Interrupt System

4.1 Interrupt Control

Interrupt Pending Register

When the hardware detects one of the eight interrupts it sets the corresponding bit in the pending interrupt register (INT_PENDING-09H). When the interrupt vector is taken, the pending bit is cleared. This register, the format of which is shown in Figure 22, can be read or modified as a byte register. It can be read to determine which of the interrupts are pending at any given time or modified to either clear pending interrupts or generate interrupts under software control. Any software which modifies the INT_PENDING register should ensure that the entire operation is indivisible. The easiest way to do this is to use the logical instructions in the two or three operand format, for example:

```
ANDB INT_PENDING,#11111101B
      ; Clears the A/D Interrupt
ORB  INT_PENDING,#00000010B
      ; Sets the A/D Interrupt
```

Caution must be used when writing to the pending register to clear interrupts. If the interrupt has already been acknowledged when the bit is cleared, a 4 state time "partial" interrupt cycle will occur. This is because the 8096BH will have to fetch the next instruction of the normal instruction flow, instead of proceeding with the interrupt processing as it was going to. The effect on the program will be essentially that of an extra NOP. This can be prevented by clearing the bits using a 2 operand immediate logical, as the 8096BH holds off acknowledging interrupts during these "read/modify/write" instructions.

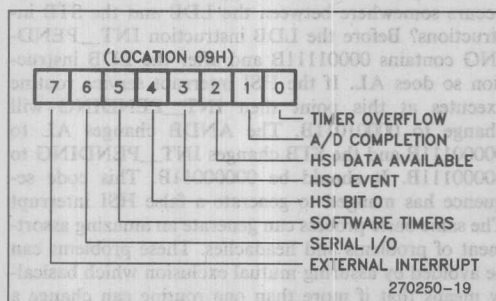


Figure 22. Interrupt Pending Register

Interrupt Mask Register

Individual interrupts can be enabled or disabled by setting or clearing bits in the interrupt mask register (INT_MASK-08H). The format of this register is the same as that of the Interrupt Pending Register shown in Figure 22.

The INT_MASK register can be read or written as byte register. A one in any bit position will enable the corresponding interrupt source and a zero will disable the source. The hardware will save any interrupts that occur by setting bits in the pending register, even if the interrupt mask bit is cleared. The INT_MASK register also can be accessed as the lower eight bits of the PSW so the PUSHF and POPF instructions save and restore the INT_MASK register as well as the global interrupt lockout and the arithmetic flags.

GLOBAL DISABLE

The processing of all interrupts can be disabled by clearing the I bit in the PSW. Setting the I bit will enable interrupts that have mask register bits which are set. The I bit is controlled by the EI (Enable Interrupts) and DI (Disable Interrupts) instructions. Note that the I bit only controls the actual servicing of interrupts. Interrupts that occur during periods of lockout will be held in the pending register and serviced on a prioritized basis when the lockout period ends.

4.2 Interrupt Priorities

The priority encoder looks at all of the interrupts which are both pending and enabled, and selects the one with the highest priority. The priorities are shown in Figure 20 (7 is highest, 0 is lowest). The interrupt generator then forces a call to the location in the indicated vector location. This location would be the starting location of the Interrupt Service Routine (ISR).

This priority selection controls the order in which pending interrupts are passed to the software via interrupt calls. The software can then implement its own priority structure by controlling the mask register (INT_MASK). To see how this is done, consider the case of a serial I/O service routine which must run at a priority level which is lower than the HSI data available interrupt but higher than any other source. The "preamble" and exit code for this interrupt service routine would look like this:

```
serial_io_isr:
PUSHF      ; Save the PSW
            (Includes INT_MASK)
LDB  INT_MASK,#00000100B
EI          ; Enable interrupts again
;
;
; } Service the interrupt
;
;
POPF      ; Restore the PSW
RET
```


Note that location 200CH in the interrupt vector table would have to be loaded with the value of the label `serial_io_isr` and the interrupt be enabled for this routine to execute.

There is an interesting chain of instruction side-effects which makes this (or any other) 8096 interrupt service routine execute properly:

- a) After the hardware decides to process an interrupt, it generates and executes a special interrupt-call instruction, which pushes the current program counter onto the stack and then loads the program counter with the contents of the vector table entry corresponding to the interrupt. The hardware will not allow another interrupt to be serviced immediately following the interrupt-call. This guarantees that once the interrupt-call starts, the first instruction of the interrupt service routine will execute.
- b) The `PUSHF` instruction, which is now guaranteed to execute, saves the PSW in the stack and then clears the PSW. The PSW contains, in addition to the arithmetic flags, the `INT_MASK` register and the global disable flag (I). The hardware will not allow an interrupt following a `PUSHF` instruction and, by the time the `LD` instruction starts, all of the interrupt enable flags will be cleared. Now there is guaranteed execution of the `LD INT_MASK` instruction.
- c) The `LD INT_MASK` instruction enables those interrupts that the programmer chooses to allow to interrupt the serial I/O interrupt service routine. In this example only the HSI data available interrupt will be allowed to do this but any interrupt or combination of interrupts could be enabled at this point, even the serial interrupt. It is the loading of the `INT_MASK` register which allows the software to establish its own priorities for interrupt servicing independently from those that the hardware enforces.
- d) The `EI` instruction reenables the processing of interrupts.
- e) The actual interrupt service routine executes within the priority structure established by the software.
- f) At the end of the service routine the `POPF` instruction restores the PSW to its state when the interrupt-call occurred. The hardware will not allow interrupts to be processed following a `POPF` instruction so the execution of the last instruction (`RET`) is guaranteed before further interrupts can occur. The reason that this `RET` instruction must be protected in this fashion is that it is quite likely that the `POPF` instruction will reenables an interrupt which is already pending. If this interrupt were serviced before the `RET` instruction, then the return address to the code that was executing when the original interrupt occurred would be left on the stack. While this does not present a problem to the program flow, it could result in a stack overflow if interrupts are occurring at a high frequency. The `POPF` instruction also pops the

`INT_MASK` register (part of the PSW), so any changes made to this register during a routine which ends with a `POPF` will be lost.

Notice that the "preamble" and exit code for the interrupt service routine does not include any code for saving or restoring registers. This is because it has been assumed that the interrupt service routine has been allocated its own private set of registers from the on-board register file. The availability of some 230 bytes of register storage makes this quite practical.

4.3 Critical Regions

Interrupt service routines must share some data with other routines. Whenever the programmer is coding those sections of code which access these shared pieces of data, great care must be taken to ensure that the integrity of the data is maintained. Consider clearing a bit in the interrupt pending register as part of a non-interrupt routine:

```
LDB      AL,INT_PENDING
ANDB     AL,#bit_mask
STB      AL,INT_PENDING
```

This code works if no other routines are operating concurrently, but will cause occasional but serious problems if used in a concurrent environment. (All programs which make use of interrupts must be considered to be part of a concurrent environment.) To demonstrate this problem, assume that the `INT_PENDING` register contains 00001111B and bit 3 (HSO event interrupt pending) is to be reset. The code does work for this data pattern but what happens if an HSI interrupt occurs somewhere between the `LDB` and the `STB` instructions? Before the `LDB` instruction `INT_PENDING` contains 00001111B and after the `LDB` instruction so does `AL`. If the HSI interrupt service routine executes at this point then `INT_PENDING` will change to 00001011B. The `ANDB` changes `AL` to 00000111B and the `STB` changes `INT_PENDING` to 00000111B. It should be 00000011B. This code sequence has managed to generate a false HSI interrupt. The same basic process can generate an amazing assortment of problems and headaches. These problems can be avoided by assuring mutual exclusion which basically means that if more than one routine can change a variable, then the programmer must ensure exclusive access to the variable during the entire operation on the variable.

In many cases the instruction set of the 8096 allows the variable to be modified with a single instruction. The code in the above example can be implemented with a single instruction.

```
ANDB     INT_PENDING,#bit_mask
```

Instructions are indivisible so mutual exclusion is ensured in this case. Changes to the INT_PENDING register must be made as a single instruction, since bits can be changed in this register even if interrupts are disabled. Depending on system configurations, several other SFRs might also need to be changed in a single instruction for the same reason.

When variables must be modified without interruption, and a single instruction can not be used, the programmer must create what is termed a critical region in which it is safe to modify the variable. One way to do this is to simply disable interrupts with a DI instruction, perform the modification, and then re-enable interrupts with an EI instruction. The problem with this approach is that it leaves the interrupts enabled even if they were not enabled at the start. A better solution is to enter the critical region with a PUSHF instruction which saves the PSW and also clears the interrupt enable flags. The region can then be terminated with a POPF instruction which returns the interrupt enable to the state it was in before the code sequence. It should be noted that some system configurations might require more protection to form a critical region. An example is a system in which more than one processor has access to a common resource such as memory or external I/O devices.

4.4 Interrupt Timing

Interrupts are not always acknowledged immediately. If the interrupt signal does not occur prior to 4 state-times before the end of an instruction, the interrupt will not be acknowledged until after the next instruction has been executed. This is because an instruction is fetched and prepared for execution a few state times before it is actually executed.

There are 6 instructions which always inhibit interrupts from being acknowledged until after the next instruction has been executed. These instructions are:

- EI, DI — Enable and Disable Interrupts
- POPF, PUSHF — Pop and Push Flags
- SIGND — Prefix to perform signed multiply and divide (Note that this is not an ASM-96 Mnemonic, but is used for signed multiply and divide)
- TRAP — Software interrupt

When an interrupt is acknowledged, the interrupt pending bit is cleared, and a call is forced to the location indicated by the specified interrupt vector. This call occurs after the completion of the instruction in process, except as noted above. The procedure of getting the vector and forcing the call requires 21 state times. If the stack is in external RAM an additional 3 state times are required.

The maximum number of state times required from the time an interrupt is generated (not acknowledged) until the 8096 begins executing code at the desired location is the time of the longest instruction, NORML (Normalize — 42 state times), plus the 4 state times prior to the end of the previous instruction, plus the response time (21 to 24 state times). Therefore, the maximum response time is 70 (42 + 4 + 24) state times. This does not include the 12 state times required for PUSHF if it is used as the first instruction in the interrupt routine or additional latency caused by having the interrupt masked or disabled. Refer to Figure 22A, Interrupt Response Time, to visualize an example of worst case scenario.

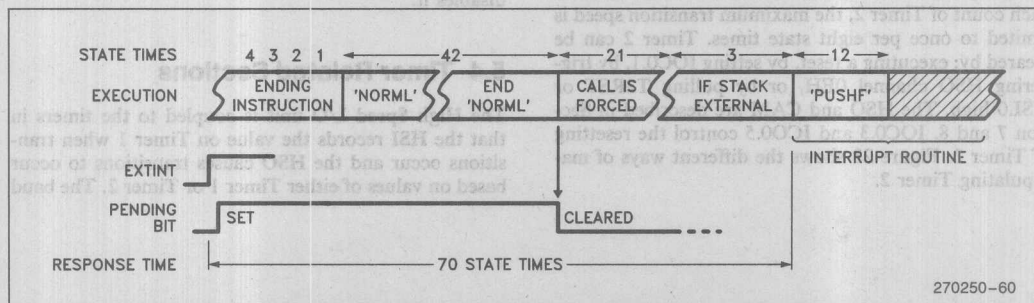


Figure 22A. Interrupt Response Time

Interrupt latency time can be reduced by careful selection of instructions in areas of code where interrupts are expected. Using 'EI' followed immediately by a long instruction (e.g. MUL, NORML, etc.) will increase the maximum latency by 4 state times, as an interrupt cannot occur between EI and the instruction following EI. The "DI", "PUSHF", "POPF" and "TRAP" instructions will also cause the same situation. Typically the PUSHF, POPF and TRAP instructions would only effect latency when one interrupt routine is already in process, as these instructions are seldom used at other times.

5.0 TIMERS

Two 16-bit timers are available for use on the 8096. The first is designated "Timer 1", the second, "Timer 2". Timer 1 is used to synchronize events to real time, while Timer 2 can be clocked externally and synchronizes events to external occurrences.

5.1 Timer 1

Timer 1 is clocked once every eight state times and can be cleared only by executing a reset. The only other way to change its value is by writing to 000CH but this is a test mode which sets both timers to 0FFFFH and should not be used in programs.

5.2 Timer 2

Timer 2 can be incremented by transitions (one count each transition, rising and falling) on either T2CLK or HSI.1. The multiple functionality of the timer is determined by the state of I/O Control Register 0, bit 7 (IOC0.7). To ensure that all CAM entries are checked each count of Timer 2, the maximum transition speed is limited to once per eight state times. Timer 2 can be cleared by: executing a reset, by setting IOC0.1, by triggering HSO channel 0EH, or by pulling T2RST or HSI.0 high. The HSO and CAM are described in Section 7 and 8. IOC0.3 and IOC0.5 control the resetting of Timer 2. Figure 23 shows the different ways of manipulating Timer 2.

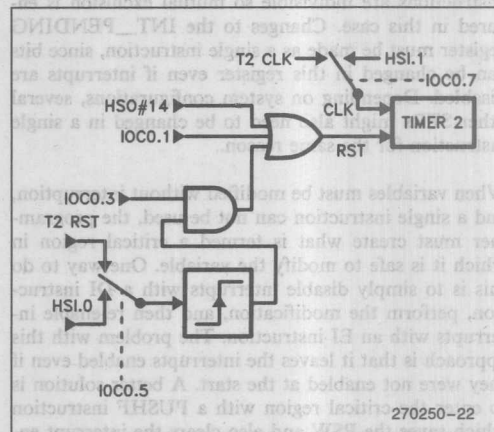


Figure 23. Timer 2 Clock and Reset Options

5.3 Timer Interrupts

Both Timer 1 and Timer 2 can be used to trigger a timer overflow interrupt and set a flag in the I/O Status Register 1 (IOS1). The interrupts are controlled by IOC1.2 and IOC1.3 respectively. The flags are set in IOS1.5 and IOS1.4, respectively.

Caution must be used when examining the flags, as any access (including Compare and Jump on Bit) of IOS1 clears bits 0 through 5 including the software timer flags. It is, therefore, recommended to write the byte to a temporary register before testing bits. The general enabling and disabling of the timer interrupts are controlled by the Interrupt Mask Register bit 0. In all cases, setting a bit enables a function, while clearing a bit disables it.

5.4 Timer Related Sections

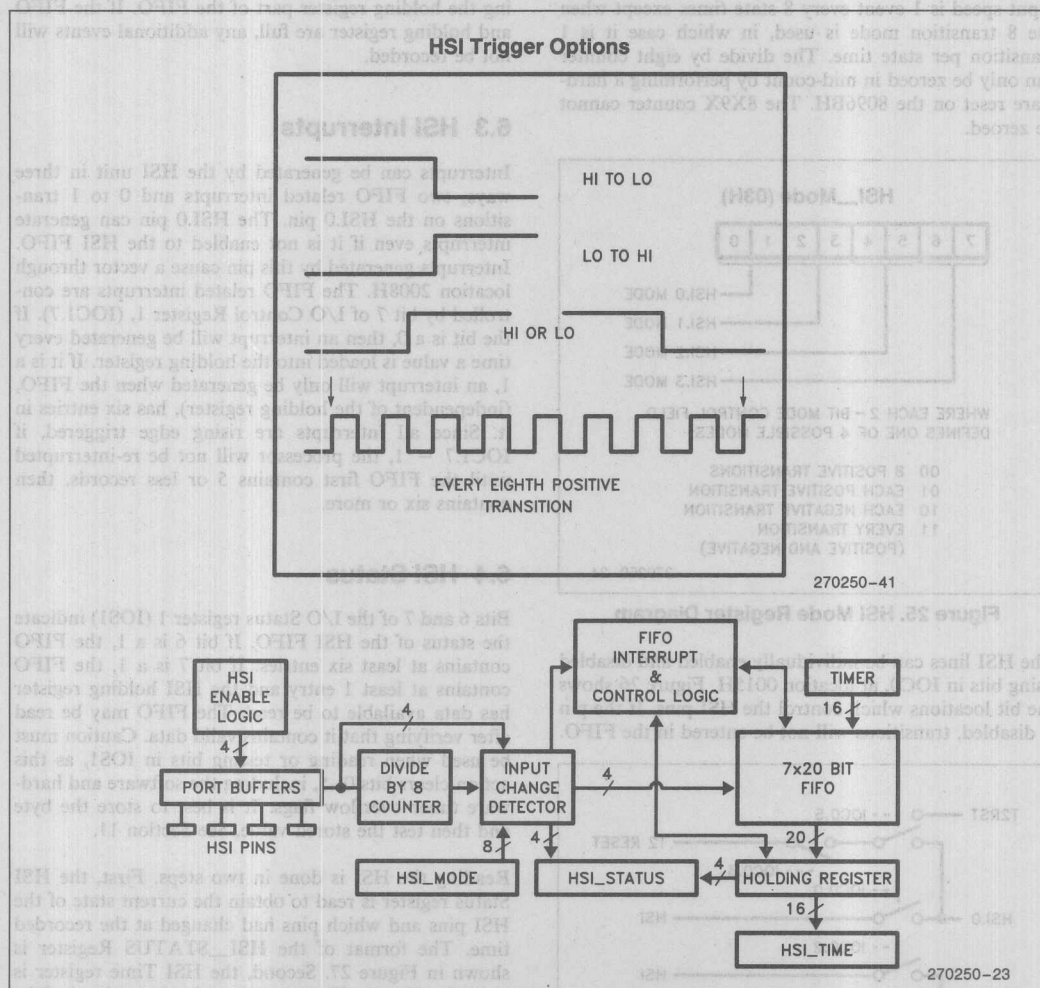
The High Speed I/O unit is coupled to the timers in that the HSI records the value on Timer 1 when transitions occur and the HSO causes transitions to occur based on values of either Timer 1 or Timer 2. The baud

rate generator can use the T2CLK pin as input to its counter. a complete listing of the functions of IOS1, IOC0, and IOC1 are in Section 11.

6.0 HIGH SPEED INPUTS

The High Speed Input Unit (HSI), can be used to record the time at which an event occurs with respect to

Timer 1. There are 4 lines (HSI.0 through HSI.3) which can be used in this mode and up to a total of 8 events can be recorded. HSI.2 and HSI.3 are bidirectional pins which can also be used as HSO.4 and HSO.5. The I/O Control Registers (IOC0 and IOC1) are used to determine the functions of these pins. A block diagram of the HSI unit is shown in Figure 24.



6.1 HSI Modes

There are 4 possible modes of operation for each of the HSI pins. The HSI mode register is used to control which pins will look for what type of events. The 8-bit register is set up as shown in Figure 25.

High and low levels each need to be held for at least 1 state time to ensure proper operation. The maximum input speed is 1 event every 8 state times except when the 8 transition mode is used, in which case it is 1 transition per state time. The divide by eight counter can only be zeroed in mid-count by performing a hardware reset on the 8096BH. The 8X9X counter cannot be zeroed.

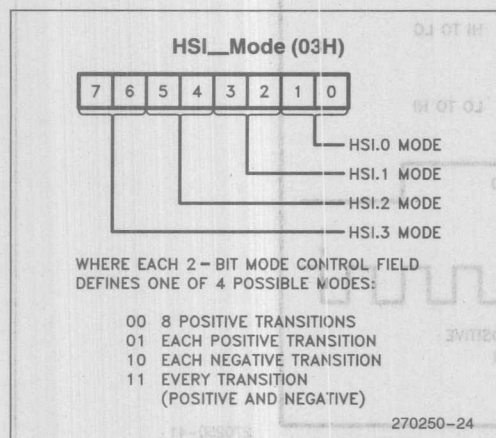


Figure 25. HSI Mode Register Diagram

The HSI lines can be individually enabled and disabled using bits in IOC0, at location 0015H. Figure 26 shows the bit locations which control the HSI pins. If the pin is disabled, transitions will not be entered in the FIFO.

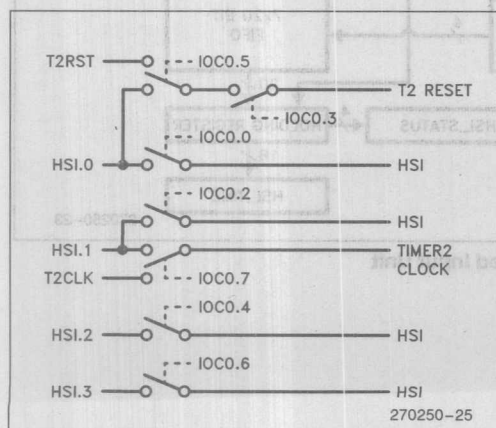


Figure 26. IOC0 Control of HSI Pin Functions

6.2 HSI FIFO

When an HSI event occurs, a 7×20 FIFO stores the 16 bits of Timer 1 and the 4 bits indicating which pins had events. It can take up to 8 state times for this information to reach the holding register. For this reason, 8 state times must be allowed between consecutive reads of HSI_TIME. When the FIFO is full, one additional event, for a total of 8 events, can be stored by considering the holding register part of the FIFO. If the FIFO and holding register are full, any additional events will not be recorded.

6.3 HSI Interrupts

Interrupts can be generated by the HSI unit in three ways; two FIFO related interrupts and 0 to 1 transitions on the HSI.0 pin. The HSI.0 pin can generate interrupts even if it is not enabled to the HSI FIFO. Interrupts generated by this pin cause a vector through location 2008H. The FIFO related interrupts are controlled by bit 7 of I/O Control Register 1, (IOC1.7). If the bit is a 0, then an interrupt will be generated every time a value is loaded into the holding register. If it is a 1, an interrupt will only be generated when the FIFO, (independent of the holding register), has six entries in it. Since all interrupts are rising edge triggered, if IOC1.7 = 1, the processor will not be re-interrupted until the FIFO first contains 5 or less records, then contains six or more.

6.4 HSI Status

Bits 6 and 7 of the I/O Status register 1 (IOS1) indicate the status of the HSI FIFO. If bit 6 is a 1, the FIFO contains at least six entries. If bit 7 is a 1, the FIFO contains at least 1 entry and the HSI holding register has data available to be read. The FIFO may be read after verifying that it contains valid data. Caution must be used when reading or testing bits in IOS1, as this action clears bits 0-5, including the software and hardware timer overflow flags. It is best to store the byte and then test the stored value. See Section 11.

Reading the HSI is done in two steps. First, the HSI Status register is read to obtain the current state of the HSI pins and which pins had changed at the recorded time. The format of the HSI_STATUS Register is shown in Figure 27. Second, the HSI Time register is read. Reading the Time register unloads one level of the FIFO, so if the Time register is read before the Status register, the event information in the Status register will be lost. The HSI Status register is at location 06H and the HSI Time registers are in locations 04H and 05H.

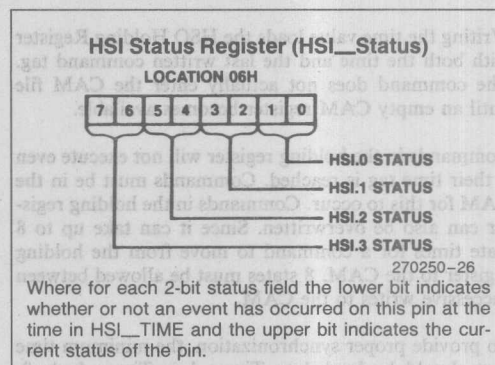
If the HSI_TIME register is read without the holding register being loaded, the returned value will be indeterminate. Under the same conditions, the four bits in

It should be noted that many of the Status register conditions are changed by a reset, see Section 13. A complete listing of the functions of IOS0, IOS1, and IOC1 can be found in Section 11.

The High Speed Output unit, (HSO), is used to trigger events at specific times with minimal CPU overhead. These events include: starting an A to D conversion, resetting Timer 2, setting 4 software flags, and switching 6 output lines (HSO.0 through HSO.5). Up to eight events can be pending at one time and interrupts can be generated whenever any of these events are triggered. HSO.4 and HSO.5 are bidirectional pins which can also be used as HSL.2 and HSL.3 respectively. Bits 4 and 6 of I/O Control Register 1, (IOC1.4, IOC1.6), enable HSO.4 and HSO.5 as outputs.

The HSO unit can generate two types of interrupts. The HSO execution interrupt (vector = (2006H)) is generated (if enabled) for HSO commands which operate one or more of the six output pins. The other HSO interrupt is the software timer interrupt (vector = (200BH)) which is generated (if enabled) by any other HSO command, (e.g. triggering the A/D, resetting Timer 2 or generating a software time delay).

A block diagram of the HSO unit is shown in Figure 28. The Content Addressable Memory (CAM) file is the center of control. One CAM register is compared with the timer values every state time, taking 8 state times to compare all CAM registers with the timers. This defines the time resolution of the HSO to be 8 state times (2.0 microseconds at an oscillator frequency of 12 MHz).



Each CAM register is 23 bits wide. Sixteen bits specify the time at which the action is to be carried out and 7 bits specify both the nature of the action and whether Timer 1 or Timer 2 is the reference. The format of the

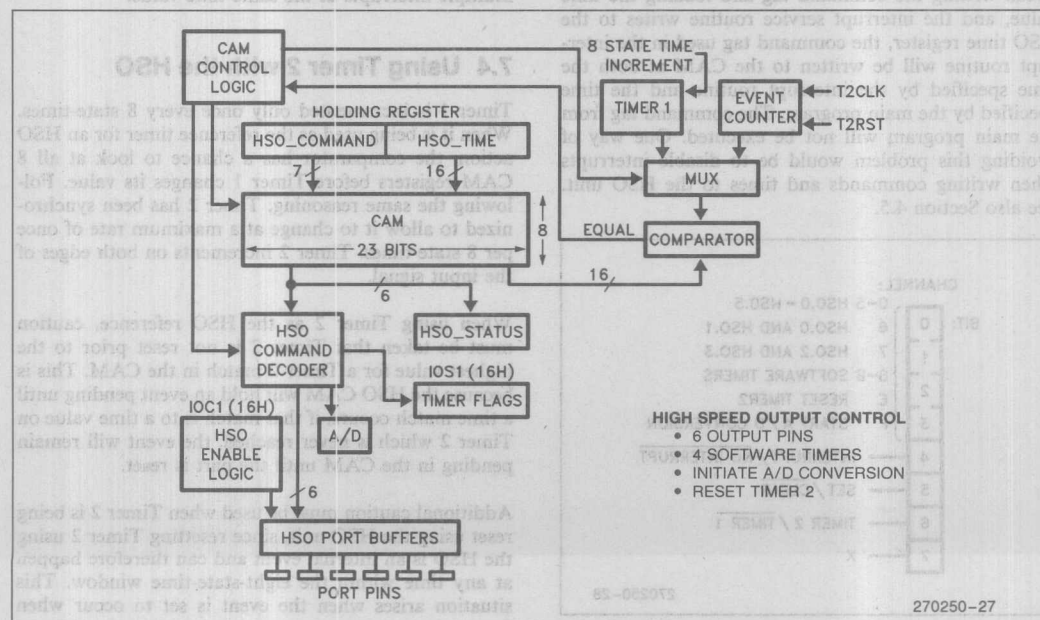


Figure 28. High Speed Output Unit

command to the HSO unit is shown in Figure 29. Note that bit 5 is ignored for command channels 8 through 0FH.

To enter a command into the CAM file, write the 7-bit "Command Tag" into location 0006H followed by the time at which the action is to be carried out into word address 0004H. The typical code would be:

```
LDB HSO_COMMAND, #what_to_do
ADD HSO_TIME, TIMER1, #when_to_do_it
```

Writing the time value loads the HSO Holding Register with both the time and the last written command tag. The command does not actually enter the CAM file until an empty CAM register becomes available.

Commands in the holding register will not execute even if their time tag is reached. Commands must be in the CAM for this to occur. Commands in the holding register can also be overwritten. Since it can take up to 8 state times for a command to move from the holding register to the CAM, 8 states must be allowed between successive writes to the CAM.

To provide proper synchronization, the minimum time that should be loaded to Timer 1 is $\text{Timer 1} + 2$. Smaller values may cause the Timer match to occur 65,636 counts later than expected. A similar restriction applies if Timer 2 is used.

Care must be taken when writing the command tag for the HSO. If an interrupt occurs during the time between writing the command tag and loading the time value, and the interrupt service routine writes to the HSO time register, the command tag used in the interrupt routine will be written to the CAM at both the time specified by the interrupt routine and the time specified by the main program. The command tag from the main program will not be executed. One way of avoiding this problem would be to disable interrupts when writing commands and times to the HSO unit. See also Section 4.5.

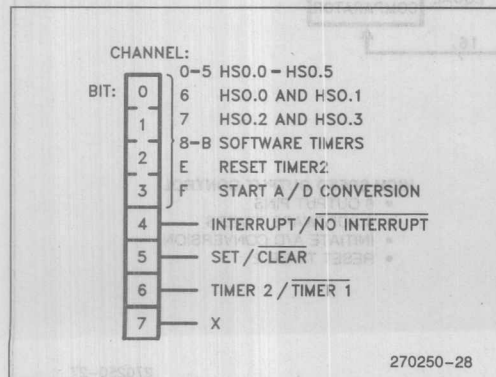


Figure 29. HSO Command Tag Format

7.2 HSO Status

Before writing to the HSO, it is desirable to ensure that the Holding Register is empty. If it is not, writing to the HSO will overwrite the value in the Holding Register. I/O Status Register 0 (IOS0) bits 6 and 7 indicate the status of the HSO unit. This register is described in Section 11. If IOS0.6 equals 0, the holding register is empty and at least one CAM register is empty. If IOS0.7 equals 0, the holding register is empty.

The programmer should carefully decide which of these two flags is the best to use for each application.

7.3 Clearing the HSO

All 8 CAM locations of the HSO are compared before any action is taken. This allows a pending external event to be cancelled by simply writing the opposite event to the CAM. However, once an entry is placed in the CAM, it cannot be removed until either the specified timer matches the written value or the chip is reset. If, as an example, a command has been issued to set HSO.1 when $\text{TIMER 1} = 1234$, then entering a second command which clears HSO.1 when $\text{TIMER 1} = 1234$ will result in no operation on HSO.1. Both commands will remain in the CAM until $\text{TIMER 1} = 1234$.

Internal events are not synchronized to Timer 1, and therefore cannot be cleared. This includes events on HSO channels 8 through F and all interrupts. Since interrupts are not synchronized it is possible to have multiple interrupts at the same time value.

7.4 Using Timer 2 with the HSO

Timer 1 is incremented only once every 8 state-times. When it is being used as the reference timer for an HSO action, the comparator has a chance to look at all 8 CAM registers before Timer 1 changes its value. Following the same reasoning, Timer 2 has been synchronized to allow it to change at a maximum rate of once per 8 state-times. Timer 2 increments on both edges of the input signal.

When using Timer 2 as the HSO reference, caution must be taken that Timer 2 is not reset prior to the highest value for a Timer 2 match in the CAM. This is because the HSO CAM will hold an event pending until a time match occurs, if that match is to a time value on Timer 2 which is never reached, the event will remain pending in the CAM until the part is reset.

Additional caution must be used when Timer 2 is being reset using the HSO unit, since resetting Timer 2 using the HSO is an internal event and can therefore happen at any time within the eight-state-time window. This situation arises when the event is set to occur when

Timer 2 is equal to zero. If HSI.0 or the T2RST pin is used to clear Timer 2, and Timer 2 equal to zero triggers the event, then the event may not occur. This is because HSI.0 and T2RST clear Timer 2 asynchronously, and Timer 2 may then be incremented to one before the HSO CAM entry can be read and acted upon. This can be avoided by setting the event to occur when Timer 2 is equal to one. This method will ensure that there is enough time for the CAM entry recognition.

The same asynchronous nature can affect events scheduled to occur at the same time as an internal Timer 2 reset. These events should be logged into the CAM with a Timer 2 value of zero. When using this method to make a programmable modulo counter, the count will stay at the maximum Timer 2 value only until the Reset T2 command is recognized. The count will stay at zero for the transition which would have changed the count from "N" to zero, and then changed to a one on the next transition.

7.5 Software Timers

The HSO can be programmed to generate interrupts at preset times. Up to four such "Software Timers" can be in operation at a time. As each preprogrammed time is reached, the HSO unit sets a Software Timer Flag. If the interrupt bit in the command tag was set then a Software Timer Interrupt will also be generated. The interrupt service routine can then examine I/O Status register 1 (IOS1) to determine which software timer expired and caused the interrupt. When the HSO resets Timer 2 or starts an A to D conversion, it can also be programmed to generate a software timer interrupt but there is no flag to indicate that this has occurred.

If more than one software timer interrupt occurs in the same time frame it is possible that multiple software timer interrupts will be generated.

Each read or test of any bit in IOS1 will clear bits 0 through 5. Be certain to save the byte before testing it unless you are only concerned with 1 bit. See also Section 11.5.

A complete listing of the functions of IOS0, IOS1, and IOC1 can be found in Section 11. The Timers are described in Section 5 and the HSI is described in Section 6.

8.0 ANALOG INTERFACE

The 8096H can easily interface to analog signals using its Analog to Digital Converter and its Pulse-Width-Modulated (PWM) output and HSO Unit. Analog inputs are accepted by the 8-input, 10-bit A to D converter. The PWM and HSO units provide digital signals which can be filtered for use as analog outputs.

8.1 Analog Inputs

A to D conversion is performed on one of the 8 inputs at a time using successive approximation with a result equal to the ratio of the input voltage divided by the analog supply voltage. If the ratio is 1.00, then the result will be all ones. The A/D converter is available on selected members of the MCS-96 family. See Section 14 for the device selection matrix.

Each conversion on the 8096BH requires 88 state-times (22 μ s at 12 MHz) independent of the accuracy desired or value of input voltage. The input voltage must be in the range of 0 to VREF, the analog reference and supply voltage. For proper operation, VREF (the reference voltage and analog power supply) must be held nominally at 5V. The A/D result is calculated from the formula:

$$1023 \times (\text{input voltage} - \text{ANGND}) / (\text{VREF} - \text{ANGND})$$

It can be seen from this formula that changes in VREF or ANGND effect the output of the converter. This can be advantageous if a ratiometric sensor is used since these sensors have an output that can be measured as a proportion of VREF.

ANGND must be tied to VSS (digital ground) in order for the 8096BH to operate properly. This common connection should be made as close to the chip as possible, and using good bulk and high frequency by-pass capacitors to decouple power supply variations and noise from the circuit. Analog design rules call for one and only one common connection between analog and digital returns to eliminate unwanted ground variations.

int₈₀₈₆

open for 4 state times which are included in the 88 state-time conversion period. The exact timings of the A/D converter can be found in Section 3 of the Hardware Design chapter.

The 8X9X devices do not have a sample and hold, so the input voltage must be held constant through the entire conversion. The conversion time is 168 state times (42 μ s at 12 MHz) on the 8X9X devices.

8.2 A/D Commands

Analog signals can be sampled by any one of the 8 analog input pins (ACH0 through ACH7) which are shared with Port 0. ACH7 can also be used as an external interrupt if IOC1.1 is set (see Sections 4 and 11). The A/D Command Register, at location 02H, selects which channel is to be converted and whether the conversion should start immediately or when the HSO (Channel #0FH) triggers it. The A/D command regis-

ter is used as the trigger. A to D commands are formatted as shown in Figure 30.

The command register is double buffered so it is possible to write a command to start a conversion triggered by the HSO while one is still in progress. Care must be taken when this is done since if a new conversion is started while one is already in progress, the conversion in progress is cancelled and the new one is started. When a conversion is started, the result register is cleared. For this reason the result register must be read before a new conversion is started or data will be lost.

8.3 A/D Results

Results of the analog conversions are read from the A/D Result Register at locations 02H and 03H. Although these addresses are on a word boundary, they must be read as individual bytes. Information in the A/D Result register is formatted as shown in Figure 31. Note that the status bit may not be set until 8 state

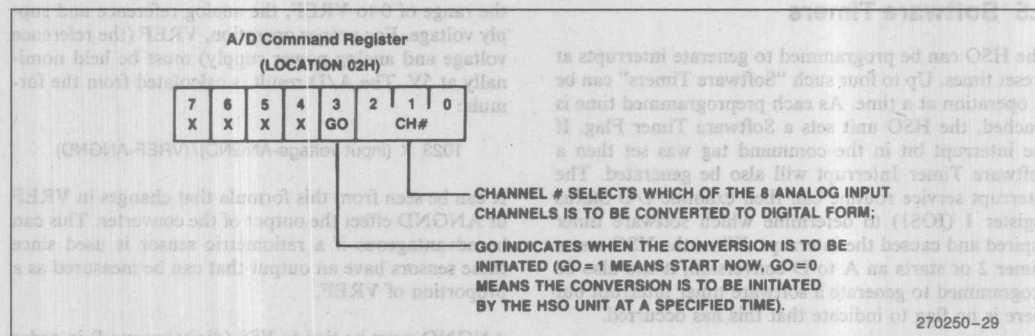


Figure 30. A/D Command Register

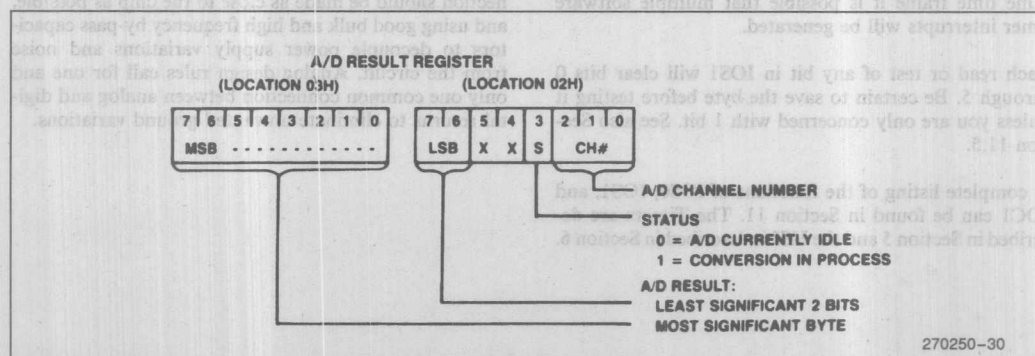


Figure 31. A/D Result Register

state times before testing it. Information on using the HSO is in Section 7.

8.4 Pulse Width Modulation Output (D/A)

Digital to analog conversion can be done with the Pulse Width Modulation output; a block diagram of the circuit is shown in Figure 32. The 8-bit counter is incremented every state time. When it equals 0, the PWM output is set to a one. When the counter matches the value in the PWM register, the output is switched low. When the counter overflows, the output is once again switched high. A typical output waveform is shown in

Figure 33. Note that when the PWM register equals 00, the output is always low. Additionally, the PWM register will only be reloaded from the temporary latch when the counter overflows. This means that the compare circuit will not recognize a new value to compare against until the counter has expired the remainder of the current 8-bit count.

The output waveform is a variable duty cycle pulse which repeats every 256 state times ($64 \mu s$ at 12 MHz). Changes in the duty cycle are made by writing to the PWM register at location 17H. There are several types of motors which require a PWM waveform for most efficient operation. Additionally, if this waveform is integrated it will produce a DC level which can be changed in 256 steps by varying the duty cycle.

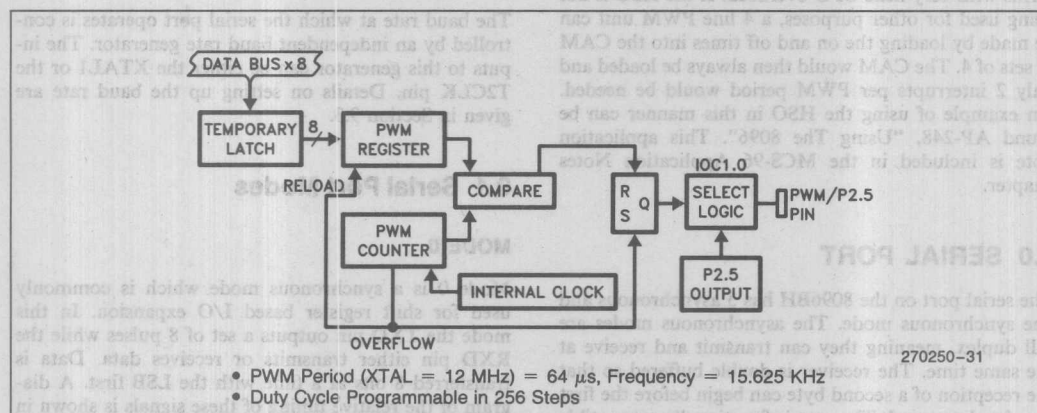


Figure 32. Pulse Width Modulated (D/A) Output

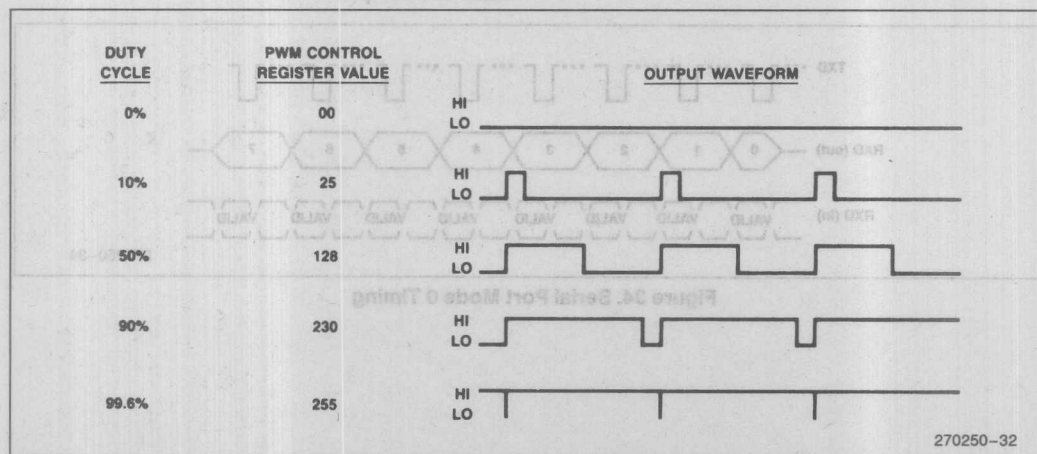


Figure 33. Typical PWM Outputs

Details about the hardware required for smooth, accurate D/A conversion can be found in Section 4 of the Hardware Design chapter. Typically, some form of buffer and integrator are needed to obtain the most usefulness from this feature.

The PWM output shares a pin with Port 2, pin 5 so that these two features cannot be used at the same time. IOC1.0 equal to 1 selects the PWM function instead of the standard port function. More information on IOC1 is in Section 11.

8.5 PWM Using the HSO

The HSO unit can be used to generate PWM waveforms with very little CPU overhead. If the HSO is not being used for other purposes, a 4 line PWM unit can be made by loading the on and off times into the CAM in sets of 4. The CAM would then always be loaded and only 2 interrupts per PWM period would be needed. An example of using the HSO in this manner can be found AP-248, "Using The 8096". This application note is included in the MCS-96 Application Notes chapter.

9.0 SERIAL PORT

The serial port on the 8096BH has 3 asynchronous and one synchronous mode. The asynchronous modes are full duplex, meaning they can transmit and receive at the same time. The receiver is double buffered so that the reception of a second byte can begin before the first byte has been read. The port is functionally compatible

with the serial port on the MCS-51 family of microcontrollers, although the software used to control the ports is different.

Control of the serial port is handled through the Serial Port Control/Status Register at location 11H. Figure 37 shows the layout of this register. The details of using it to control the serial port will be discussed in Section 9.2.

Data to and from the serial port is transferred through SBUF (rx) and SBUF (tx), both located at 07H. Although these registers share the same address, they are physically separate, with SBUF (rx) containing the data received by the serial port and SBUF (tx) used to hold data ready for transmission. The program cannot write to SBUF (rx) or read from SBUF (tx).

The baud rate at which the serial port operates is controlled by an independent baud rate generator. The inputs to this generator can be either the XTAL1 or the T2CLK pin. Details on setting up the baud rate are given in Section 9.3.

9.1 Serial Port Modes

MODE 0

Mode 0 is a synchronous mode which is commonly used for shift register based I/O expansion. In this mode the TXD pin outputs a set of 8 pulses while the RXD pin either transmits or receives data. Data is transferred 8 bits at a time with the LSB first. A diagram of the relative timing of these signals is shown in Figure 34. Note that this is the only mode which uses RXD as an output.

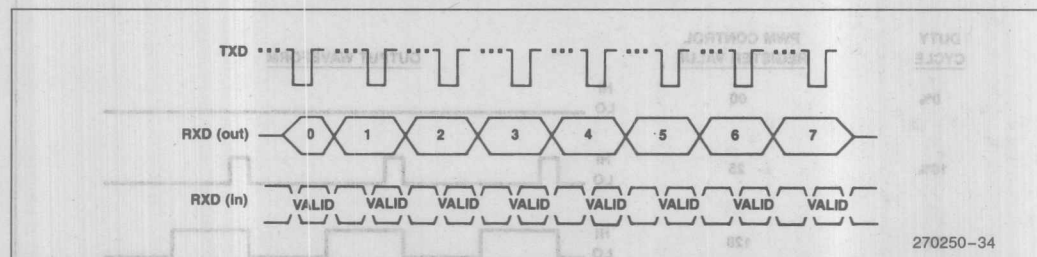


Figure 34. Serial Port Mode 0 Timing

Although it is not possible to transmit and receive at the same time using this mode, two external gates and a port pin can be used to time-multiplex the two functions. An example of multiplexing transmit and receive is discussed in Section 6.1 of the Hardware Design chapter.

MODE 1

Mode 1 is the standard asynchronous communications mode. The data frame used in this mode is shown in Figure 35. It consists of 10 bits; a start bit (0), 8 data bits (LSB first), and a stop bit (1). If parity is enabled, (the PEN bit is set to a 1), an even parity bit is sent instead of the 8th data bit and parity is checked on reception.

MODE 2

Mode 2 is the asynchronous 9th bit recognition mode. This mode is commonly used with Mode 3 for multi-processor communications. Figure 36 shows the data frame used in this mode. It consists of a start bit (0), 9 data bits (LSB first), and a stop bit (1). When transmitting, the 9th bit can be set to a one by setting the TB8 bit in the control register before writing to SBUF (tx). The TB8 bit is cleared on every transmission, so it must be set prior to writing to SBUF (tx) each time it is desired. During reception, the serial port interrupt and the Receive Interrupt (RI) bit will not be set unless the 9th bit being received is set. This provides an easy way to have selective reception on a data link. Parity cannot be enabled in this mode.

MODE 3

Mode 3 is the asynchronous 9th bit mode. The data frame for this mode is identical to that of Mode 2. The transmission differences between Mode 3 and Mode 2 are that parity can be enabled (PEN=1) and cause the 9th data bit to take the even parity value. The TB8 bit can still be used if parity is not enabled (PEN=0). When in Mode 3, a reception always causes an interrupt, regardless of the state of the 9th bit. The 9th bit is stored if PEN=0 and can be read in bit RB8. If PEN=1 then RB8 becomes the Receive Parity Error (RPE) flag.

9.2 Controlling the Serial Port

Control of the serial port is done through the Serial Port Control (SP_CON) and Serial Port Status (SP_STAT) registers shown in Figure 37. Writing to location 11H accesses SP_CON while reading it accesses SP_STAT. Note that reads of SP_STAT will return indeterminate data in the lower 5 bits and writing to the upper 3 bits of SP_CON has no effect on chip functionality. The TB8 bit is cleared after each transmission and both TI and RI are cleared whenever SP_STAT (not SP_CON) is accessed. Whenever the TXD pin is used for the serial port it must be enabled by setting IOC1.5 to a 1. IOC1 is discussed further in Section 11.3. Information on the hardware connections and timing of the serial port is in Section 6 of the Hardware Design chapter.

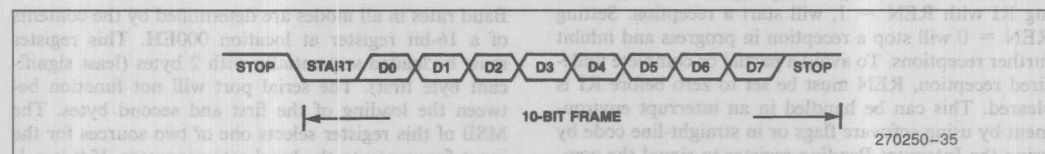


Figure 35. Serial Port Frame—Mode 1

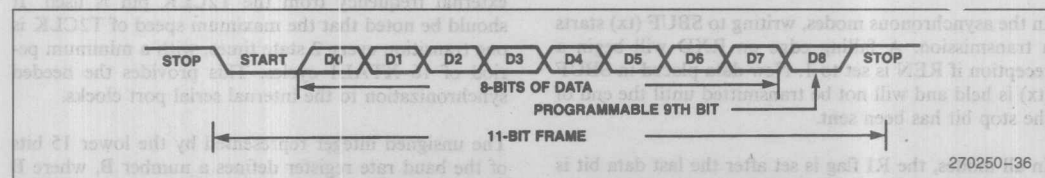


Figure 36. Serial Port Frame Modes 2 and 3

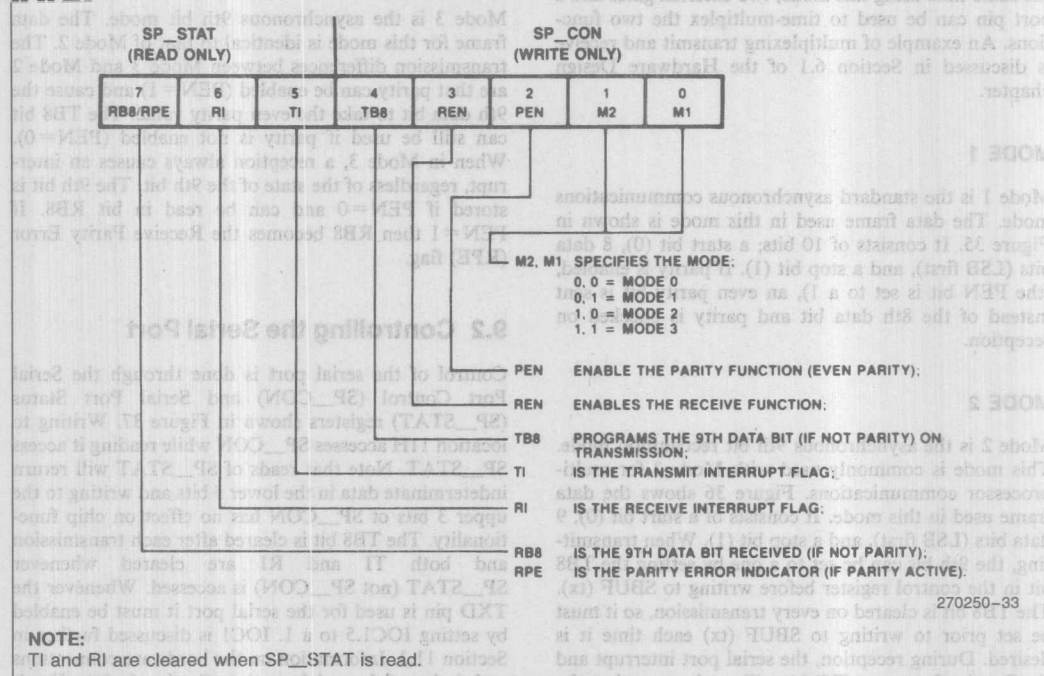


Figure 37. Serial Port Control/Status Register

In Mode 0, if REN = 1, writing to SBUF (tx) will start a transmission. Causing a rising edge on REN, or clearing RI with REN = 1, will start a reception. Setting REN = 0 will stop a reception in progress and inhibit further receptions. To avoid a partial or complete undesired reception, REN must be set to zero before RI is cleared. This can be handled in an interrupt environment by using software flags or in straight-line code by using the Interrupt Pending register to signal the completion of a reception.

In the asynchronous modes, writing to SBUF (tx) starts a transmission. A falling edge on RXD will begin a reception if REN is set to 1. New data placed in SBUF (tx) is held and will not be transmitted until the end of the stop bit has been sent.

In all modes, the RI flag is set after the last data bit is sampled approximately in the middle of the bit time. Also for all modes, the TI flag is set after the last data bit (either 8th or 9th) is sent, also in the middle of the bit time. The flags clear when SP_STAT is read, but do not have to be clear for the port to receive or transmit. The serial port interrupt bit is set as a logical OR of the RI and TI bits. Note that changing modes will reset the Serial Port and abort any transmission or reception in progress on the channel.

9.3 Determining Baud Rates

Baud rates in all modes are determined by the contents of a 16-bit register at location 000EH. This register must be loaded sequentially with 2 bytes (least significant byte first). The serial port will not function between the loading of the first and second bytes. The MSB of this register selects one of two sources for the input frequency to the baud rate generator. If it is a 1, the frequency on the XTAL1 pin is selected, if not, the external frequency from the T2CLK pin is used. It should be noted that the maximum speed of T2CLK is one transition every 2 state times, with a minimum period of 16 XTAL1 cycles. This provides the needed synchronization to the internal serial port clocks.

The unsigned integer represented by the lower 15 bits of the baud rate register defines a number B, where B has a maximum value of 32767. The baud rate for the four serial modes using either XTAL1 or T2CLK as the clock source is given by:

Using XTAL1:

$$\text{Mode 0: Baud Rate} = \frac{\text{XTAL1 frequency}}{4 * (B + 1)}; \quad B \neq 0$$

Others: $\text{Baud Rate} = \frac{\text{XTAL1 frequency}}{64 * (B + 1)}$

Using T2CLK:

Mode 0: $\text{Baud Rate} = \frac{\text{T2CLK frequency}}{B}$; $B \neq 0$

Others: $\text{Baud Rate} = \frac{\text{T2CLK frequency}}{16 * B}$; $B \neq 0$

Note that B cannot equal 0, except when using XTAL1 in other than mode 0.

Common baud rate values, using XTAL1 at 12 MHz, are shown below.

Baud Rate	Baud Register Value	
	Mode 0	Others
9600	8137H	8013H
4800	8270H	8026H
2400	84E1H	804DH
1200	89C3H	809BH
300	A70FH	8270H

The maximum baud rates are 1.5 Mbaud synchronous and 187.5 Kbaud asynchronous with 12 MHz on XTAL1.

9.4 Multiprocessor Communications

Mode 2 and 3 are provided for multiprocessor communications. In Mode 2 if the received 9th data bit is not 1, the serial port interrupt is not activated. The way to use this feature in multiprocessor systems is described below.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address frame which identifies the target slave. An address frame will differ from a data frame in that the 9th data bit is 1 in an address frame and 0 in a data frame. No slave in Mode 2 will be interrupted by a data frame. An address frame, however, will interrupt all slaves so that each slave can examine the received byte and see if it is being addressed. The addressed slave switches to Mode 3 to receive the coming data frames, while the slaves that were not addressed stay in Mode 2 and go on about their business.

10.0 I/O PORTS

There are five 8-bit I/O ports on the 8096. Some of these ports are input only, some are output only, some

are bidirectional and some have alternate functions. In addition to these ports, the HSI/O unit can be used to provide extra I/O lines if the timer related features of these lines are not needed.

Input ports connect to the internal bus through an input buffer. Output ports connect through an output buffer to an internal register that hold the bits to be output. Bidirectional ports consist of an internal register, an input buffer, and an output buffer.

Port 0 is an input port which is also used as the analog input for the A to D converter. Port 1 is a quasi-bidirectional port. Port 2 contains three types of port lines: quasi-bidirectional, input and output. The input and output lines are shared with other functions in the 8096BH as shown in Table 4. Ports 3 and 4 are open-drain bidirectional ports which share their pins with the address/data bus.

Table 4. Port 2 Alternate Functions

Port	Function	Alternate Function	Controlled by
P2.0	Output	TXD (Serial Port Transmit)	IOC1.5
P2.1	Input	RXD (Serial Port Receive)	N/A
P2.2	Input	EXTINT (External Interrupt)	IOC1.1
P2.3	Input	T2CLK (Timer 2 Input)	IOC0.7
P2.4	Input	T2RST (Timer 2 Reset)	IOC0.5
P2.5	Output	PWM (Pulse-Width Modulation)	IOC1.0
P2.6	Quasi-Bidirectional		
P2.7	Quasi-Bidirectional		

Section 2 of the Hardware Design chapter contains additional information on the timing, drive capabilities, and input impedances of I/O pins.

10.1 Input Ports

Input ports and pins can only be read. There are no output drivers on these pins. The input leakage of these pins is in the microamp range. The specific values can be found in the data sheet for the device being considered.

In addition to acting as a digital input, each line of Port 0 can be selected to be the input of the A to D converter as discussed in Section 8. The pins on Port 0 are tested

to have D.C. leakage of 3 microamps or less, as specified in the data sheet for the device being considered. The capacitance on these pins is approximately 5 pF and will instantaneously increase by around 5 pF when the pin is being sampled by the A to D converter.

The 8096BH samples the input to the A/D for 4 state times at the beginning of the conversion. The 8X9X devices sample the A/D pin 10 times during a conversion. Details on the A to D converter can be found in Section 8 of this chapter and in Section 3 of the Hardware Design chapter.

10.2 Quasi-Bidirectional Ports

Port 1, Port 2.6 and Port 2.7 are quasi-bidirectional ports. "Quasi-bidirectional" means that the port pin has a weak internal pullup that is always active and an internal pulldown which can be on to output a 0, or off to output a 1. If the internal pulldown is left off (by writing a 1 to the pin), the pin's logic level can be controlled by an external pulldown. If the external pulldown is on, it will input a 0 to the 8096BH, if it is off, a 1 will be input. From the user's point of view, the main difference between a quasi-bidirectional port and a standard input port is that the quasi-bidirectional port will source current if externally pulled low. It will also pull itself high if left unconnected.

In parallel with the weak internal pullup is a much stronger internal pullup that is activated for one state time when the pin is internally driven from 0 to 1. This is done to speed up the 0-to-1 transition time. When this pullup is on the pin can typically source 30 milliamps to V_{SS} .

When the processor writes to the pins of a quasi-bidirectional port it actually writes into a register which in turn drives the port pin. When the processor reads these ports, it senses the status of the pin directly. If a port pin is to be used as an input then the software should write a one to its associated SFR bit, this will cause the low-impedance pull-down device to turn off and leave the pin pulled up with a relatively high im-

pedance pullup device which can be easily driven down by the device driving the input.

If some pins of a port are to be used as inputs and some are to be used as outputs the programmer should be careful when writing to the port.

Particular care should be exercised when using XOR opcodes or any opcode which is a read-modify-write instruction. It is possible for a Quasi-Bidirectional Pin to be written as a one, but read back as a zero if an external device (i.e., a transistor base) is pulling the pin below V_{IH} . See the Hardware Design Chapter Section 2.2 for further details on using the Quasi-Bidirectional Ports.

10.3 Output Ports

Output pins include the bus control lines, the HSO lines, and some of Port 2. These pins can only be used as outputs as there are no input buffers connected to them. It is not possible to use immediate logical instructions such as XOR PORT2, #00111B to toggle these pins. The output currents on these ports is higher than that of the quasi-bidirectional ports.

10.4 Ports 3 and 4/AD0-15

These pins have two functions. They are either bidirectional ports with open-drain outputs or System Bus pins which the memory controller uses when it is accessing off-chip memory. If the \overline{EA} line is low, the pins always act as the System Bus. Otherwise they act as bus pins only during a memory access. If these pins are being used as ports and bus pins, ones must be written to them prior to bus operations.

Accessing Port 3 and 4 as I/O is easily done from internal registers. Since the LD and ST instructions require the use of internal registers, it may be necessary to first move the port information into an internal location before utilizing the data. If the data is already internal, the LD is unnecessary. For instance, to write a word value to Port 3 and 4...

```
LD intreg, portdata ; register ← data
; not needed if already internal
```

```
ST intreg, 1FFEh ; register → Port 3 and 4
```

To read Port 3 and 4 requires that "ones" be written to the port registers to first setup the input port configuration circuit. Note that the ports are reset to this input condition, but if zeroes have been written to the port, then ones must be re-written to any pins which are to be used as inputs. Reading Port 3 and 4 from a previously written zero condition is as follows ...

```
LD intregA, #0FFFFH ; setup port change mode pattern
```

```
ST intregA, 1FFEh ; register → Port 3 and 4
; LD & ST not needed if previously
; written as ones
```

```
LD intregB, 1FFEh ; register ← Port 3 and 4
```

Note that while the format of the LD and ST instructions are similar, the source and destination directions change.

When acting as the system bus the pins have strong drivers to both V_{CC} and V_{SS} . These drivers are used whenever data is being output on the system bus and are not used when data is being output by Ports 3 and 4. Only the pins and input buffers are shared between the bus and the ports. The ports use different output buffers which are configured as open-drain, and require pullup resistors. (open-drain is the MOS version of open-collector.) The port pins and their system bus functions are shown in Table 5.

Table 5. P3,4/AD0-15 Pins

Port Pin	System Bus Function
P3.0	AD0
P3.1	AD1
P3.2	AD2
P3.3	AD3
P3.4	AD4
P3.5	AD5
P3.6	AD6
P3.7	AD7
P4.0	AD8
P4.1	AD9
P4.2	AD10
P4.3	AD11
P4.4	AD12
P4.5	AD13
P4.6	AD14
P4.7	AD15

11.0 STATUS AND CONTROL REGISTERS

There are two I/O Control registers, IOC0 and IOC1. IOC0 controls Timer 2 and the HSI lines. IOC1 controls some pin functions, interrupt sources and 2 HSO pins.

Whenever input lines are switched between two sources, or enabled, it is possible to generate transitions on these lines. This could cause problems with respect to edge sensitive lines such as the HSI lines, Interrupt line, and Timer 2 control lines.

11.1 I/O Control Register 0 (IOC0)

IOC0 is located at 0015H. The four HSI lines can be enabled or disabled to the HSI unit by setting or clearing bits in IOC0. Timer 2 functions including clock and reset sources are also determined by IOC0. The control bit locations are shown in Figure 38.

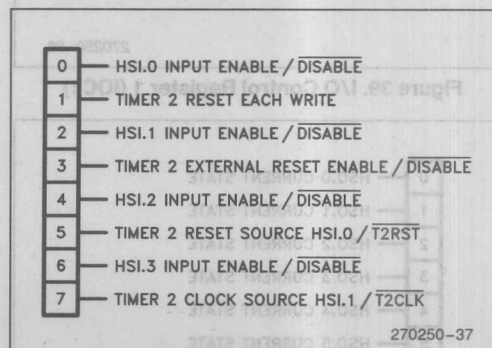


Figure 38. I/O Control Register 0 (IOC0)

int 1

IOC1 is used to select some pin functions and enable or disable some interrupt sources. Its location is 0016H. Port pin P2.5 can be selected to be the PWM output instead of a standard output. The external interrupt source can be selected to be either EXTINT (same pin as P2.2) or Analog Channel 7 (ACH7, same pin as P0.7). Timer 1 and Timer 2 overflow interrupts can be individually enabled or disabled. The HSI interrupt can be selected to activate either when there is 1 FIFO entry or 7. Port pin P2.0 can be selected to be the TXD output. HSO.4 and HSO.5 can be enabled or disabled to the HSO unit. More information on interrupts is available in Section 4. The positions of the IOC1 control bits are shown in Figure 39.

11.3 I/O Status Register 0 (IOS0)

There are two I/O Status registers, IOS0 and IOS1. IOS0, located at 0015H, holds the current status of the HSO lines and CAM. The status bits of IOS0 are shown in Figure 40.

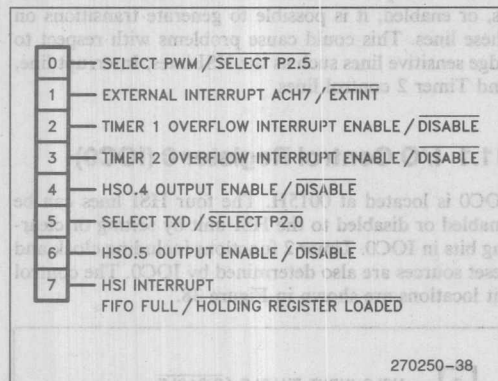


Figure 39. I/O Control Register 1 (IOC1)

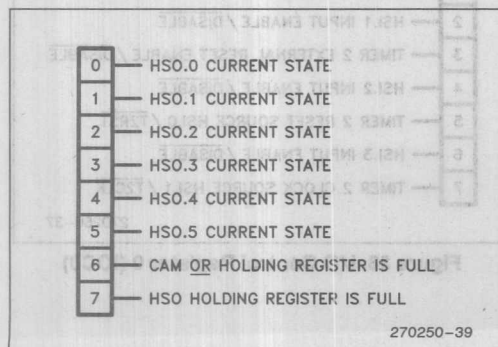
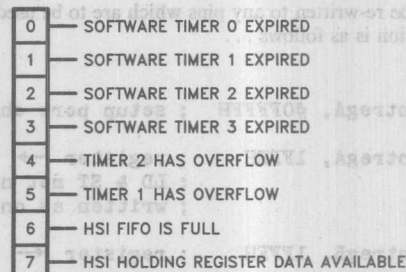


Figure 40. I/O Status Register 0 (IOS0)



270250-40

Figure 41. HSI Status Register 1 (IOS1)

11.4 I/O Status Register 1 (IOS1)

IOS1 is located at 016H. It contains status bits for the timers and the HSI/O. The positions of these bits are shown in Figure 41.

Whenever the processor reads this register all of the time-related flags (bits 5 through 0) are cleared. This applies not only to explicit reads such as:

```
LDB AL, IOS1
```

but also to implicit reads such as:

```
JB IOS1.3, somewhere_else
```

which jumps to somewhere_else if bit 3 of IOS1 is set. In most cases this situation can best be handled by having a byte in the register file which is used to maintain an image of lower five bits of the register. Any time a hardware timer interrupt or a HSO software timer interrupt occurs the byte can be updated:

```
ORB IOS1_image, IOS1
```

leaving IOS1_image containing all the flags that were set before plus all the new flags that were read and cleared from IOS1. Any other routine which needs to sample the flags can safely check IOS1_image. Note that if these routines need to clear the flags that they have acted on, then the modification of IOS1_image must be done from inside a critical region (see Section 4.4).

12.0 WATCHDOG TIMER

The WatchDog Timer (WDT) provides a means to recover gracefully from a software upset. When the watchdog is enabled it will initiate a hardware reset unless the software clears it every 64K state times.

The WDT is implemented as an 8-bit timer with an 8-bit prescaler. The prescaler is not synchronized, so the timer will overflow between 65280 and 65535 state times after being reset. When the timer overflows it pulls down the RESET pin for at least two state times, resetting the 8096BH and any other devices tied to the RESET line. If a large capacitor is connected to the line, the pin may take a long time to go low. This will effect the length of time the pin is low and the voltage on the pin when it is finished falling. Section 1.4 of the Hardware Design chapter contains more information about reset hardware connections.

The WDT is enabled the first time it is cleared. Once it is enabled, it can only be disabled by resetting the 8096BH. The internal bit which controls the watchdog can typically maintain its state through power glitches as low as V_{SS} and as high as 7.0V for up to one millisecond.

The 8X9X devices do not have the extra glitch protection on the WDT enable bit.

Enabling and clearing the WDT is done by writing a "01EH" followed by a "0E1H" to the WDT register at location 0AH. This double write is used to help prevent accidental clearing of the timer.

12.1 Software Protection Hints

Glitches and noise on the PC board can cause software upsets, typically by changing either memory locations or the program counter. These changes can be internal to the chip or be caused by bad data returning to the chip.

There are both hardware and software solutions to noise problems, but the best solution is good design practice and a few ounces of prevention. The software can be designed so that the watchdog times out if the program does not progress properly. The watchdog will also time-out if the software error was due to ESD (Electrostatic Discharge) or other hardware related problems. This prevents the controller from having a malfunction for longer than 16 milliseconds if a 12 MHz oscillator is used.

When using the WDT to protect software it is desirable to reset it from only one place in code. This will lessen the chance that an undesired WDT reset will occur. The section of code that resets the WDT should monitor the other code sections for proper operation. This

can be done by checking variables to make sure they are within reasonable values. Simply using a software timer to reset the WDT every 15 milliseconds will not provide much protection against minor problems.

It is also recommended that unused areas of code be filled with NOPs and periodic jumps to an error routine or RST (reset chip) instructions. This is particularly important in the code around lookup tables, since if lookup tables are executed undesired results will occur. Wherever space allows, each table should be surrounded by 7 NOPs (the longest 8096 instruction has 7 bytes) and a RST or jump to error routine instruction. Since RST is a one-byte instruction, the NOPs are not needed if RSTs are used instead of jumps to an error routine. This will help to ensure a speedy recovery should the processor have a glitch in the program flow. Since RST instruction has an opcode of 0FFH, pulling the data lines high with resistors will cause an RST to be executed if unimplemented memory is addressed.

12.2 Disabling The Watchdog

The watchdog should be disabled by software not initializing it. If this is not possible, such as during program development, the watchdog can be disabled by holding the RESET pin at 2.0V to 2.5V. Voltages over 2.5V on the pin could quickly damage the part. Even at 2.5V, using this technique for other than debugging purposes is not recommended, as it may effect long term reliability. It is further recommended that any part used in this way for more than several seconds, not be used in production versions of products. Section 1.6 of the Hardware Design chapter has more information on disabling the Watchdog Timer.

13.0 RESET

13.1 Reset Signal

As with all processors, the 8096BH must be reset each time the power is turned on. This is done by holding the RESET pin low for at least 2 state times after the power supply is within tolerance and the oscillator has stabilized.

On 8X9X devices the RESET pin must be held low long enough for the power supply, oscillator and back-bias generator to stabilize. Typically, the back-bias generator requires one millisecond to stabilize.

After the RESET pin is brought high, a ten state reset sequence is executed. During this time, the Chip Configuration Byte (CCB) is read from location 2018H and written to the 8096BH Chip Configuration Register (CCR). If the voltage on the EA pin selects the inter-

nal/external execution mode the CCB is read from internal ROM/EPROM. If the voltage on the \overline{EA} pin selects the external execution only mode the CCB is read from external memory.

The 8096BH can be reset using a capacitor, 1-shot, or any other method capable of providing a pulse of at least 2 state times longer than required for V_{CC} and the oscillator to stabilize.

For best functionality, it is suggested that the reset pin be pulled low with an open collector device. In this way, several reset sources can be wire ORed together. Remember, the \overline{RESET} pin itself can be a reset source when the RST instruction is executed or when the Watchdog Timer overflows. Details of hardware suggestions for reset can be found in Section 1.4 of the Hardware Design chapter.

13.2 Reset Status

The I/O lines and control lines of the 8096BH will be in their reset state within 2 state times after reset is low, with V_{CC} and the oscillator stabilized. Prior to that time, the status of the I/O lines is indeterminate. After the 10 state time reset sequence, the Special Function Registers will be set as follows:

Register	Reset Value
Port 1	11111111B
Port 2	110XXXX1B
Port 3	11111111B
Port 4	11111111B
PWM Control	00H
Serial Port (Transmit)	undefined
Serial Port (Receive)	undefined
Baud Rate Register	undefined
Serial Port Control	XXXX0XXXB
Serial Port Status	X00XXXXXB
A/D Command	undefined
A/D Result	undefined
Interrupt Pending	undefined
Interrupt Mask	00000000B
Timer 1	0000H
Timer 2	0000H
Watchdog Timer	0000H
HSI Mode	11111111B
HSI Status	undefined
IOS0	00000000B
IOS1	00000000B
IOC0	X0X0X0X0B
IOC1	X0X0XXX1B
HSI FIFO	empty
HSO CAM	empty
HSO lines	0000000B
PSW	0000H
Stack Pointer	undefined
Program Counter	2080H

Figure 42. Register Reset Status

Other conditions following a reset are:

Pin	Reset Value
\overline{RD}	high
$\overline{WR}/\overline{WRL}$	high
$\overline{ALE}/\overline{ADV}$	high
$\overline{BHE}/\overline{WRH}$	low
INST	high
\overline{ALE} (8X9X)	low

Figure 43. Bus Control Pins Reset Status

It is important to note that the Stack Pointer and Interrupt Pending Register are undefined, and need to be initialized in software. The Interrupts are disabled by both the mask register and PSW.9 after a reset.

13.3 Reset Sync Mode

The \overline{RESET} line can be used to start the 8096BH at an exact state time to provide for synchronization of test equipment and multiple chip systems. \overline{RESET} is active low. To synchronize parts, \overline{RESET} is brought high on the rising edge of XTAL1. Complete details on synchronizing parts can be found in Section 1.5 of the Hardware Design chapter.

It is very possible that parts which start in sync may not stay that way. The best example of this would be when a "jump on I/O bit" is being used to hold the processor in a loop. If the line changes during the time it is being tested, one processor may see it as a one, while the other sees it as a zero. The result is that one processor will do an extra loop, thus putting it several states out of sync with the other.

14.0 QUICK REFERENCE

14.1 Pin Description

On the 48-pin parts the following pins are not bonded out: Port1, Port0 (Analog In) bits 0-3, T2CLK (P2.3), T2RST (P2.4), P2.6, P2.7, CLKOUT, INST, NMI, BUSWIDTH (TEST on 8X9X devices).

PIN DESCRIPTIONS

(PIN DESCRIPTIONS (Continued))

Symbol	Name and Function
V _{CC}	Main supply voltage (5V).
V _{SS}	Digital circuit ground (0V).
V _{PD}	RAM standby supply voltage (5V). This voltage must be present during normal operation. In a Power Down condition (i.e. V _{CC} drops to zero), if RESET is activated before V _{CC} drops below spec and V _{PD} continues to be held within spec., the top 16 bytes in the Register File will retain their contents. RESET must be held low during the Power Down and should not be brought high until V _{CC} is within spec and the oscillator has stabilized. See Section 2.3.
V _{REF}	Reference voltage for the A/D converter (5V). V _{REF} is also the supply voltage to the analog portion of the A/D converter and the logic used to read Port 0. See Section 8.
ANGND	Reference ground for the A/D converter. Should be held at nominally the same potential as V _{SS} . See Section 8.
V _{PP} V _{BB} (8X9X)	Programming voltage for the EPROM parts. It should be +12.75V when programming and will float to 5V otherwise. It should not be above 5.5V on other than EPROM parts. This pin is V _{BB} on 8X9X parts. Systems that have this pin connected to ANGND through a capacitance (required on 8X9X parts) do not need to change.
XTAL1	Input of the oscillator inverter and of the internal clock generator. See Section 1.5.
XTAL2	Output of the oscillator inverter. See Section 1.5.
CLKOUT	Output of the internal clock generator. The frequency of CLKOUT is 1/3 the oscillator frequency. It has a 33% duty cycle. See Section 1.5.
RESET	Reset input to the chip. Input low for at least 2 state times to reset the chip. The subsequent low-to-high transition re-synchronizes CLKOUT and commences a 10-state-time sequence in which the PSW is cleared, a byte read from 2018H loads CCR, and a jump to location 2080H is executed. Input high for normal operation. RESET has an internal pullup. (The read from 2018H is not done on 8X9X parts). See Section 13.
BUSWIDTH TEST(8X9X)	Input for buswidth selection. If CCR bit 1 is a one, this pin selects the bus width for the bus cycle in progress. If BUSWIDTH is a 1, a 16-bit bus cycle occurs. If BUSWIDTH is a 0 an 8-bit cycle occurs. If CCR bit 1 is a 0, the bus is always an 8-bit bus. This pin is the TEST pin on 8X9X parts. Systems with TEST tied to V _{CC} do not need to change. If this pin is left unconnected, it will rise to V _{CC} . See Section 2.7.
NMI	A positive transition causes a vector to external memory location 0000H. External memory from 00H through 0FFH is reserved for Intel development systems.
INST	Output high during an external memory read indicates the read is an instruction fetch. INST is valid throughout the bus cycle.
EA	Input for memory select (External Access). EA equal to a TTL-high causes memory accesses to locations 2000H through 3FFFH to be directed to on-chip ROM/EPROM. EA equal to a TTL-low causes accesses to these locations to be directed to off-chip memory. EA = +12.5V causes execution to begin in the Programming mode on EPROM parts. EA has an internal pulldown, so it goes to 0 unless driven otherwise.
ALE/ADV	Address Latch Enable or Address Valid output, as selected by CCR. Both pin options provide a latch to demultiplex the address from the address/data bus. When the pin is ADV, it goes inactive high at the end of the bus cycle. ADV can be used as a chip select for external memory. ALE/ADV is activated only during external memory accesses. (The ADV function is not available on 8X9X parts). See Section 2.7.
RD	Read signal output to external memory. RD is activated only during external memory reads.

PIN DESCRIPTIONS (Continued)

Symbol	Name and Function
WR/WRL	Write and Write Low output to external memory, as selected by the CCR. $\overline{\text{WR}}$ will go low for every external write, while WRL will go low only for external writes where an even byte is being written. $\overline{\text{WR}}$ /WRL is activated only during external memory writes. (The WRL function is not available on 8X9X parts). See Section 2.7.
BHE/WRH	Bus High Enable or Write High output to external memory, as selected by the CCR. $\text{BHE} = 0$ selects the bank of memory that is connected to the high byte of the data bus. $\text{A0} = 0$ selects the bank of memory that is connected to the low byte of the data bus. Thus accesses to a 16-bit wide memory can be to the low byte only ($\text{A0} = 0$, $\text{BHE} = 1$), to the high byte only ($\text{A0} = 1$, $\text{BHE} = 0$), or both bytes ($\text{A0} = 0$, $\text{BHE} = 0$). If the WRH function is selected, the pin will go low if the bus cycle is writing to an odd memory location. (The WRH function is not available on 8X9X parts). See Section 2.7.
READY	Ready input to lengthen external memory cycles, for interfacing to slow or dynamic memory, or for bus sharing. If the pin is high, CPU operation continues in a normal manner. If the pin is low prior to the falling edge of CLKOUT, the Memory Controller goes into a wait mode until the next positive transition in CLKOUT occurs with READY high. The bus cycle can be lengthened by up to 1 μs . When the external memory is not being used, READY has no effect. Internal control of the number of wait states inserted into a bus cycle held not ready is available through configuration of CCR. READY has a weak internal pullup, so it goes to 1 unless externally pulled low. (Internal control of the number of wait states is not available on 8X9X parts). See Section 2.7.
HSI	Inputs to High Speed Input Unit. Four HSI pins are available: HSI.0, HSI.1, HSI.2, and HSI.3. Two of them (HSI.2 and HSI.3) are shared with the HSO Unit. The HSI pins are also used as inputs by EPROM parts in Programming mode. See Section 6.
HSO	Outputs from High Speed Output Unit. Six HSO pins are available: HSO.0, HSO.1, HSO.2, HSO.3, HSO.4, and HSO.5. Two of them (HSO.4 and HSO.5) are shared with the HSI Unit. See Section 7.
Port 0	8-bit high impedance input-only port. These pins can be used as digital inputs and/or as analog inputs to the on-chip A/D converter. These pins are also a mode input to EPROM parts in the Programming mode. See Section 10.
Port 1	8-bit quasi-bidirectional I/O port. See Section 10.
Port 2	8-bit multi-functional port. Six of its pins are shared with other functions in the 8096BH, the remaining 2 are quasi-bidirectional. These pins are also used to input and output control signals on EPROM parts in Programming Mode. See Section 10.
Ports 3 and 4	8-bit bi-directional I/O ports with open drain outputs. These pins are shared with the multiplexed address/data bus which has strong internal pullups. Ports 3 and 4 are also used as a command, address and data path by EPROM parts operating in the programming mode. See Sections 2.7 and 10.
ADV	Address Latch Enable or Address Valid output, as selected by CCR. Both pin options provide a latch to demultiplex the address from the address/data bus. When the pin is ADV, it goes inactive high at the end of the bus cycle. ADV can be used as a chip select for external memory. $\overline{\text{ADV}}$ is activated only during external memory accesses. (The ADV function is not available on 8X9X parts). See Section 2.7.
RD	Read signal output to external memory. RD is activated only during external memory reads.

The following is a list of pins in alphabetical order. Where a pin has two names it has been listed under both names, except for the system bus pins, AD0-AD15, which are listed under Port 3 and Port 4.

Name	68-Pin PLCC	68-Pin PGA	48-Pin DIP	name	PLCC	PGA	DIP
ACH0/P0.0	6	4	—	P1.6	31	47	—
ACH1/P0.1	5	5	—	P1.7	32	46	—
ACH2/P0.2	7	3	—	P2.0/TXD/NER	18	60	2
ACH3/P0.3	4	6	—	P2.1/RXD/PALE	17	61	1
ACH4/P0.4/MOD.0	11	67	43	P2.2/EXTINT	15	63	47
ACH5/P0.5/MOD.1	10	68	42	P2.3/T2CLK	44	34	—
ACH6/P0.6/MOD.2	8	2	40	P2.4/T2RST	42	36	—
ACH7/P0.7/MOD.3	9	1	41	P2.5/PWM/PDO	39	39	13
ALE/ADV	62	16	34	P2.6	33	45	—
ANGND	12	66	44	P2.7	38	40	—
BHE/WRH	41	37	15	P3.0/AD0	60	18	32
BUSWIDTH (TEST)	64	14	—	P3.1/AD1	59	19	31
CLKOUT	65	13	—	P3.2/AD2	58	20	30
EA	2	8	39	P3.3/AD3	57	21	29
EXTINT/P2.2/PROG	15	63	47	P3.4/AD4	56	22	28
HSI.0	24	54	3	P3.5/AD5	55	23	27
HSI.1	25	53	4	P3.6/AD6	54	24	26
HSI.2/HSO.4	26	52	5	P3.7/AD7	53	25	25
HSI.3/HSO.5	27	51	6	P4.0/AD8	52	26	24
HSO.0	28	50	7	P4.1/AD9	51	27	23
HSO.1	29	49	8	P4.2/AD10	50	28	22
HSO.2	34	44	9	P4.3/AD11	49	29	21
HSO.3	35	43	10	P4.4/AD12	48	30	20
HSO.4/HSI.2	26	52	5	P4.5/AD13	47	31	19
HSO.5/HSI.3	27	51	6	P4.6/AD14	46	32	18
INST	63	15	—	P4.7/AD15	45	33	17
NMI	3	7	—	RD	61	17	33
PWM/P2.5/PDO	39	39	13	READY	43	35	16
PALE/P2.1/RXD	17	61	1	RESET	16	62	48
PROG/P2.2/EXTINT	15	63	47	RXD/P2.1	17	61	1
PVER/P2.0/TXD	18	60	2	SALE/PVER/P2.0	18	60	2
P0.0/ACH0	6	4	—	SPROG/PDO/P2.5	39	39	13
P0.1/ACH1	5	5	—	TXD/P2.0	18	60	2
P0.2/ACH2	7	3	—	T2CLK/P2.3	44	34	—
P0.3/ACH3	4	6	—	T2RST/P2.4	42	36	—
P0.4/ACH4/MOD.0	11	67	43	VBB	37	41	12
P0.5/ACH5/MOD.1	10	68	42	VCC	1	9	38
P0.6/ACH6/MOD.2	8	2	40	VPD	14	64	46
P0.7/ACH7/MOD.3	9	1	41	VREF	13	65	45
P1.0	19	59	—	VSS	68	10	11
P1.1	20	58	—	VSS	36	42	37
P1.2	21	57	—	WR/WRL	40	38	14
P1.3	22	56	—	WRH/BHE	41	37	15
P1.4	23	55	—	XTAL1	67	11	36
P1.5	30	48	—	XTAL2	66	12	35

The Following pins are not bonded out in the 48-pin package:

P1.0 through P1.7, P0.0 through P0.3, P2.3, P2.4, P2.6, P2.7 CLKOUT, INST, NMI, TEST, T2CLK (P2.3), T2RST (P2.4).

14.3 Packaging

The MCS-96 products are available in 48-pin and 68-pin packages, with and without A/D, and with and without on-chip ROM or EPROM. The MCS-96 numbering system is shown below. Section 14.4 shows the pinouts for the 48- and 68-pin packages. The 48-pin version is offered in a Dual-In-Line package while the 68-pin versions come in a Plastic Leaded Chip Carrier (PLCC), a Pin Grid Array (PGA) or a Type "B" Leadless Chip Carrier.

The MCS®-96 Family Nomenclature

		Without A/D	With A/D
ROMless 809XBH	48 Pin		C8095BH - Ceramic DIP P8095BH - Plastic DIP
	68 Pin	A8096BH - Ceramic PGA N8096BH - PLCC	A8097BH - Ceramic PGA N8097BH - PLCC
ROM 839XBH	48 Pin		C8395BH - Ceramic DIP P8395BH - Plastic DIP
	68 Pin	A8396BH - Ceramic PGA N8396BH - PLCC	A8397BH - Ceramic PGA N8397BH - PLCC
EPROM 879XBH	48 Pin		C8795BH - Ceramic DIP
	68 Pin	A8796BH - Ceramic PGA R8796BH - Ceramic LCC	A8797BH - Ceramic PGA R8797BH - Ceramic LCC
ROMless 8096	48 Pin		C8095-90 - Ceramic DIP P8095-90 - Plastic DIP
	68 Pin	A8096-90 - Ceramic PGA N8096-90 - PLCC	A8097-90 - Ceramic PGA N8097-90 - PLCC
ROM 8396	48 Pin		C8395-90 - Ceramic DIP P8395-90 - Plastic DIP
	68 Pin	A8396-90 - Ceramic PGA N8396-90 - PLCC	A8397-90 - Ceramic PGA N8397-90 - PLCC

Transistor Count

Device Type	# MOS Gates
839X/879X	120,000
809X	50,000

MTBF Calculations*

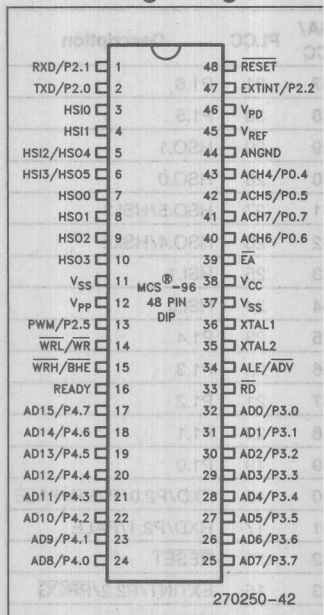
3.8×10^7 Device Hours @ 55°C
1.7×10^7 Device Hours @ 70°C

*MTBF data was obtained through calculations based upon the actual average junction temperatures under stress at 55°C and 70°C ambient.

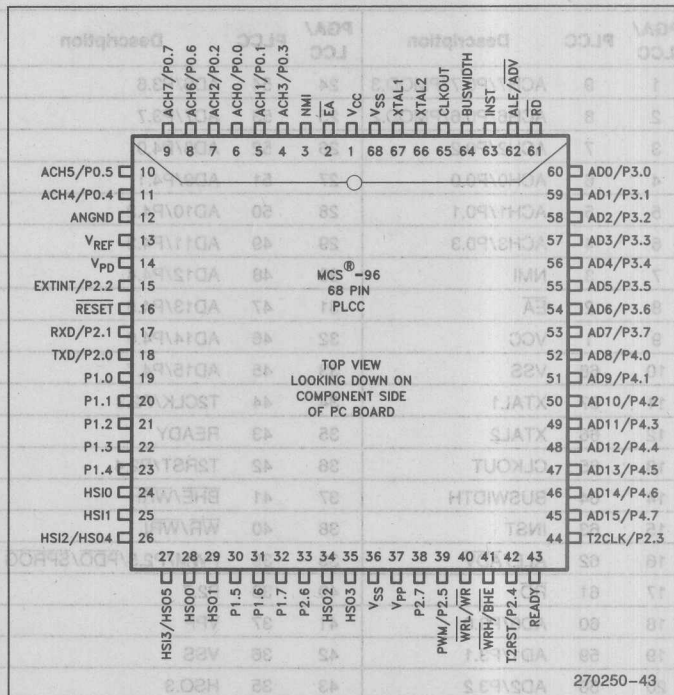
Thermal Characteristics

T _{CASE}		Package Type	θ _{Ja}	θ _{Jc}
COMM'L	EXPRESS			
85°C	100°C	PGA	35°C/W	10°C/W
85°C	100°C	PLCC	37°C/W	10°C/W
		LCC	28°C/W	—
		Plastic DIP	38°C/W	—
79.75°C	94.75°C	Ceramic DIP	26°C/W	6.5°C/W

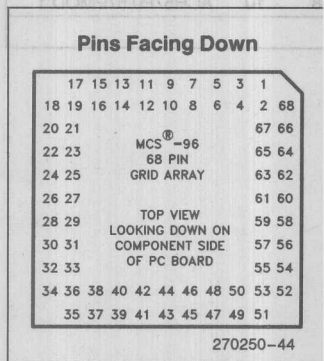
14.4 Package Diagrams



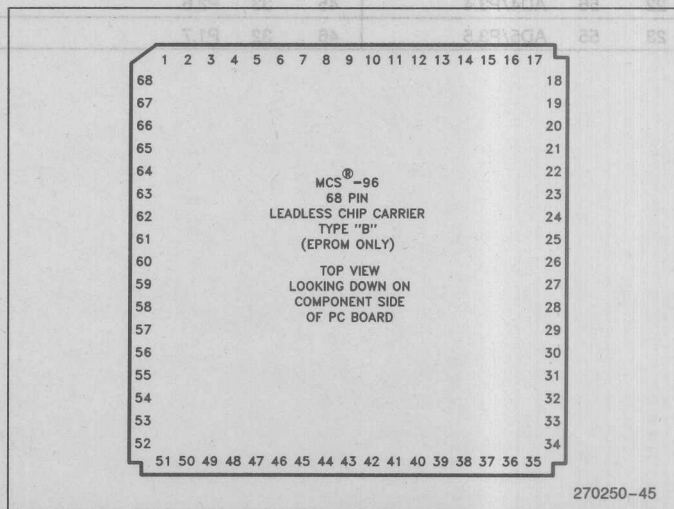
48-Pin Package



68-Pin Package (PLCC - Top View)

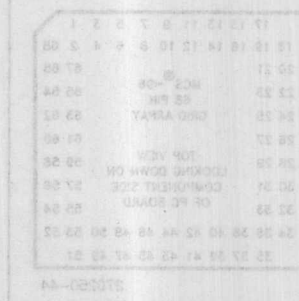
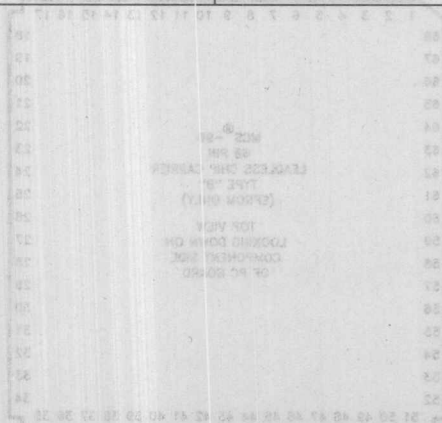


68-Pin Package
(Pin Grid Array - Top View)



68-Pin Package (LCC - Top View)

PGA/ LCC	PLCC	Description	PGA/ LCC	PLCC	Description	PGA/ LCC	PLCC	Description
1	9	ACH7/P0.7/PMOD.3	24	54	AD6/P3.6	47	31	P1.6
2	8	ACH6/P0.6/PMOD.2	25	53	AD7/P3.7	48	30	P1.5
3	7	ACH2/P0.2	26	52	AD8/P4.0	49	29	HSO.1
4	6	ACH0/P0.0	27	51	AD9/P4.1	50	28	HSO.0
5	5	ACH1/P0.1	28	50	AD10/P4.2	51	27	HSO.5/HSI.3
6	4	ACH3/P0.3	29	49	AD11/P4.3	52	26	HSO.4/HSI.2
7	3	NMI	30	48	AD12/P4.4	53	25	HSI.1
8	2	EA	31	47	AD13/P4.5	54	24	HSI.0
9	1	VCC	32	46	AD14/P4.6	55	23	P1.4
10	68	VSS	33	45	AD15/P4.7	56	22	P1.3
11	67	XTAL1	34	44	T2CLK/P2.3	57	21	P1.2
12	66	XTAL2	35	43	READY	58	20	P1.1
13	65	CLKOUT	36	42	T2RST/P2.4	59	19	P1.0
14	64	BUSWIDTH	37	41	BHE/WRH	60	18	TXD/P2.0/PVER/SALE
15	63	INST	38	40	WR/WRL	61	17	RXD/P2.1/PALE
16	62	ALE/ADV	39	39	PWM/P2.5/PDO/SPROG	62	16	RESET
17	61	RD	40	38	P2.7	63	15	EXTINT/P2.2/PROG
18	60	AD0/P3.0	41	37	VPP	64	14	VPD
19	59	AD1/P3.1	42	36	VSS	65	13	VREF
20	58	AD2/P3.2	43	35	HSO.3	66	12	ANGND
21	57	AD3/P3.3	44	34	HSO.2	67	11	ACH4/P0.4/PMOD.0
22	56	AD4/P3.4	45	33	P2.6	68	10	ACH5/P0.5/PMOD.1
23	55	AD5/P3.5	46	32	P1.7			



0FFH	POWER-DOWN RAM	255	
0F0H		240	
0EFH	INTERNAL REGISTER FILE (RAM)	239	
1AH		26	
19H	STACK POINTER	25	
18H		24	
17H		23	
16H	IOS1	22	
15H	IOS0	21	
14H		20	
13H	RESERVED	19	
12H		18	
11H	SP_STAT	17	
10H	IO PORT 2	16	
0FH	IO PORT 1	15	
0EH	IO PORT 0	14	
0DH	TIMER2 (HI)	13	
0CH	TIMER2 (LO)	12	
0BH	TIMER1 (HI)	11	
0AH	TIMER1 (LO)	10	
09H	INT_PENDING	9	
08H	INT_MASK	8	
07H	SBUF (RX)	7	
06H	HSL_STATUS	6	
05H	HSL_TIME (HI)	5	
04H	HSL_TIME (LO)	4	
03H	AD_RESULT (HI)	3	
02H	AD_RESULT (LO)	2	
01H	RO (HI)	1	
00H	RO (LO)	0	
	(WHEN READ)	(WHEN WRITTEN)	
			270250-5

NOTE:

*Registers marked by an asterisk are not present on 8X9X devices

14.7 Instruction Summary

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	$D \leftarrow D + A$	✓	✓	✓	✓	↑	—	
ADD/ADDB	3	$D \leftarrow B + A$	✓	✓	✓	✓	↑	—	
ADDC/ADDCB	2	$D \leftarrow D + A + C$	↓	✓	✓	✓	↑	—	
SUB/SUBB	2	$D \leftarrow D - A$	✓	✓	✓	✓	↑	—	
SUB/SUBB	3	$D \leftarrow B - A$	✓	✓	✓	✓	↑	—	
SUBC/SUBCB	2	$D \leftarrow D - A + C - 1$	↓	✓	✓	✓	↑	—	
CMP/CMPB	2	$D - A$	✓	✓	✓	✓	↑	—	
MUL/MULU	2	$D, D + 2 \leftarrow D * A$	—	—	—	—	—	?	2
MUL/MULU	3	$D, D + 2 \leftarrow B * A$	—	—	—	—	—	?	2
MULB/MULUB	2	$D, D + 1 \leftarrow D * A$	—	—	—	—	—	?	3
MULB/MULUB	3	$D, D + 1 \leftarrow B * A$	—	—	—	—	—	?	3
DIVU	2	$D \leftarrow (D, D + 2) / A, D + 2 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	2
DIVUB	2	$D \leftarrow (D, D + 1) / A, D + 1 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	3
DIV	2	$D \leftarrow (D, D + 2) / A, D + 2 \leftarrow \text{remainder}$	—	—	—	?	↑	—	
DIVB	2	$D \leftarrow (D, D + 1) / A, D + 1 \leftarrow \text{remainder}$	—	—	—	?	↑	—	
AND/ANDB	2	$D \leftarrow D \text{ and } A$	✓	✓	0	0	—	—	
AND/ANDB	3	$D \leftarrow B \text{ and } A$	✓	✓	0	0	—	—	
OR/ORB	2	$D \leftarrow D \text{ or } A$	✓	✓	0	0	—	—	
XOR/XORB	2	$D \leftarrow D \text{ (excl. or) } A$	✓	✓	0	0	—	—	
LD/LDB	2	$D \leftarrow A$	—	—	—	—	—	—	
ST/STB	2	$A \leftarrow D$	—	—	—	—	—	—	
LDBSE	2	$D \leftarrow A; D + 1 \leftarrow \text{SIGN}(A)$	—	—	—	—	—	—	3, 4
LDBZE	2	$D \leftarrow A; D + 1 \leftarrow 0$	—	—	—	—	—	—	3, 4
PUSH	1	$SP \leftarrow SP - 2; (SP) \leftarrow A$	—	—	—	—	—	—	
POP	1	$A \leftarrow (SP); SP \leftarrow SP + 2$	—	—	—	—	—	—	
PUSHF	0	$SP \leftarrow SP - 2; (SP) \leftarrow \text{PSW};$ $\text{PSW} \leftarrow 0000\text{H}$ $I \leftarrow 0$	0	0	0	0	0	0	
POPF	0	$\text{PSW} \leftarrow (SP); SP \leftarrow SP + 2;$ $I \leftarrow \text{PSW}$	✓	✓	✓	✓	✓	✓	
SJMP	1	$PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LJMP	1	$PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5
BR [indirect]	1	$PC \leftarrow (A)$	—	—	—	—	—	—	
SCALL	1	$SP \leftarrow SP - 2; (SP) \leftarrow PC;$ $PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LCALL	1	$SP \leftarrow SP - 2; (SP) \leftarrow PC;$ $PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5
RET	0	$PC \leftarrow (SP); SP \leftarrow SP + 2$	—	—	—	—	—	—	
J (conditional)	1	$PC \leftarrow PC + 8\text{-bit offset (if taken)}$	—	—	—	—	—	—	5
JC	1	Jump if C = 1	—	—	—	—	—	—	5
JNC	1	Jump if C = 0	—	—	—	—	—	—	5
JE	1	Jump if Z = 1	—	—	—	—	—	—	5

NOTES:

1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the Register File; A can be located anywhere in memory.
2. D, D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D, D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to a word.
5. Offset is a 2's complement number.

Mnemonic	Oper- ands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
JNE	1	Jump if Z = 0	—	—	—	—	—	—	5
JGE	1	Jump if N = 0	—	—	—	—	—	—	5
JLT	1	Jump if N = 1	—	—	—	—	—	—	5
JGT	1	Jump if N = 0 and Z = 0	—	—	—	—	—	—	5
JLE	1	Jump if N = 1 or Z = 1	—	—	—	—	—	—	5
JH	1	Jump if C = 1 and Z = 0	—	—	—	—	—	—	5
JNH	1	Jump if C = 0 or Z = 1	—	—	—	—	—	—	5
JV	1	Jump if V = 1	—	—	—	—	—	—	5
JNV	1	Jump if V = 0	—	—	—	—	—	—	5
JVT	1	Jump if VT = 1; Clear VT	—	—	—	—	0	—	5
JNVT	1	Jump if VT = 0; Clear VT	—	—	—	—	0	—	5
JST	1	Jump if ST = 1	—	—	—	—	—	—	5
JNST	1	Jump if ST = 0	—	—	—	—	—	—	5
JBS	3	Jump if Specified Bit = 1	—	—	—	—	—	—	5, 6
JBC	3	Jump if Specified Bit = 0	—	—	—	—	—	—	5, 6
DJNZ	1	D ← D - 1; if D ≠ 0 then PC ← PC + 8-bit offset	—	—	—	—	—	—	5
DEC/DECB	1	D ← D - 1	✓	✓	✓	✓	↑	—	
NEG/NEGB	1	D ← 0 - D	✓	✓	✓	✓	↑	—	
INC/INCB	1	D ← D + 1	✓	✓	✓	✓	↑	—	
EXT	1	D ← D; D + 2 ← Sign (D)	✓	✓	0	0	—	—	2
EXTB	1	D ← D; D + 1 ← Sign(D)	✓	✓	0	0	—	—	3
NOT/NOTB	1	D ← Logical Not (D)	✓	✓	0	0	—	—	
CLR/CLRB	1	D ← 0	1	0	0	0	—	—	
SHL/SHLB/SHLL	2	C ← msb ———— lsb ← 0	✓	?	✓	✓	↑	—	7
SHR/SHRB/SHRL	2	0 → msb ———— lsb → C	✓	?	✓	0	—	✓	7
SHRA/SHRAB/SHRAL	2	msb → msb ———— lsb → C	✓	✓	✓	0	—	✓	7
SETC	0	C ← 1	—	—	1	—	—	—	
CLRC	0	C ← 0	—	—	0	—	—	—	
CLRVT	0	VT ← 0	—	—	—	—	0	—	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	—	—	—	—	—	—	
EI	0	Enable All Interrupts (I ← 1)	—	—	—	—	—	—	
NOP	0	PC ← PC + 1	—	—	—	—	—	—	
SKIP	0	PC ← PC + 2	—	—	—	—	—	—	
NORML	2	Left shift till msb = 1; D ← shift count	✓	?	0	—	—	—	7
TRAP	0	SP ← SP - 2; (SP) ← PC PC ← (2010H)	—	—	—	—	—	—	9

NOTES:

1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B and A must conform to the alignment rules for the required operand type. D and B are locations in the Register File; A can be located anywhere in memory.
5. Offset is a 2's complement number.
6. Specified bit is one of the 2048 bits in the register file.
7. The "L" (Long) suffix indicates double-word operation.
8. Initiates a Reset by pulling RESET low. Software should re-initialize all the necessary registers with code starting at 2080H.
9. The assembler will not accept this mnemonic.

14.8 Opcode and State Time Listing

MNEMONIC	OPERANDS	DIRECT			IMMEDIATE			INDIRECT®				INDEXED®					
		OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE TIMES	NORMAL		AUTO-INC.		SHORT		LONG			
								OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE TIMES	OPCODE
ARITHMETIC INSTRUCTIONS																	
ADD	2	64	3	4	65	4	5	66	3	6/11	3	7/12	67	4	6/11	5	7/12
ADD	3	44	4	5	45	5	6	46	4	7/12	4	8/13	47	5	7/12	6	8/13
ADDB	2	74	3	4	75	3	4	76	3	6/11	3	7/12	77	4	6/11	5	7/12
ADDB	3	54	4	5	55	4	5	56	4	7/12	4	8/13	57	5	7/12	6	8/13
ADDC	2	A4	3	4	A5	4	5	A6	3	6/11	3	7/12	A7	4	6/11	5	7/12
ADDCB	2	B4	3	4	B5	3	4	B6	3	6/11	3	7/12	B7	4	6/11	5	7/12
SUB	2	68	3	4	69	4	5	6A	3	6/11	3	7/12	6B	4	6/11	5	7/12
SUB	3	48	4	5	49	5	6	4A	4	7/12	4	8/13	4B	5	7/12	6	8/13
SUBB	2	78	3	4	79	3	4	7A	3	6/11	3	7/12	7B	4	6/11	5	7/12
SUBB	3	58	4	5	59	4	5	5A	4	7/12	4	8/13	5B	5	7/12	6	8/13
SUBC	2	A8	3	4	A9	4	5	AA	3	6/11	3	7/12	AB	4	6/11	5	7/12
SUBCB	2	B8	3	4	B9	3	4	BA	3	6/11	3	7/12	BB	4	6/11	5	7/12
CMP	2	88	3	4	89	4	5	8A	3	6/11	3	7/12	8B	4	6/11	5	7/12
CMPB	2	98	3	4	99	3	4	9A	3	6/11	3	7/12	9B	4	6/11	5	7/12
MULU	2	6C	3	25	6D	4	26	6E	3	27/32	3	28/33	6F	4	27/32	5	28/33
MULU	3	4C	4	26	4D	5	27	4E	4	28/33	4	29/34	4F	5	28/33	6	29/34
MULUB	2	7C	3	17	7D	3	17	7E	3	19/24	3	20/25	7F	4	19/24	5	20/25
MULUB	3	5C	4	18	5D	4	18	5E	4	20/25	4	21/26	5F	5	20/25	6	21/26
MUL	2	②	4	29	②	5	30	②	4	31/36	4	32/37	②	5	31/36	6	32/37
MUL	3	②	5	30	②	6	31	②	5	32/37	5	33/38	②	6	32/37	7	33/38
MULB	2	②	4	21	②	4	21	②	4	23/28	4	24/29	②	5	23/28	6	24/29
MULB	3	②	5	22	②	5	22	②	5	24/29	5	25/30	②	6	24/29	7	25/30
DIVU	2	8C	3	25	8D	4	26	8E	3	28/32	3	29/33	8F	4	28/32	5	29/33
DIVUB	2	9C	3	17	9D	3	17	9E	3	20/24	3	21/25	9F	4	20/24	5	21/25
DIV	2	②	4	29	②	5	30	②	4	32/36	4	33/37	②	5	32/36	6	33/37
DIVB	2	②	4	21	②	4	21	②	4	24/28	4	25/29	②	5	24/28	6	25/29

270250-46

- NOTES:**
- *Long indexed and Indirect + instructions have identical opcodes with Short indexed and Indirect modes, respectively. The second byte of instructions using any Indirect or indexed addressing mode specifies the exact mode used. If the second byte is even, use Indirect or Short indexed. If it is odd, use Indirect + or Long indexed. In all cases the second byte of the instruction always specifies an even (word) location for the address referenced.
 - ① Number of state times shown for internal/external operands.
 - ② The opcodes for signed multiply and divide are the opcodes for the unsigned functions with an "FE" appended as a prefix.
 - ③ State times shown for 16-bit bus.

MNEMONIC	OPERANDS	DIRECT			IMMEDIATE			INDIRECT ^①				INDEXED ^②					
		OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE TIMES	NORMAL		AUTO-INC.		SHORT		LONG			
								OPCODE	BYTES	STATE ^③ TIMES	BYTES	STATE ^③ TIMES	OPCODE	BYTES	STATE ^③ TIMES ^④	BYTES	STATE ^③ TIMES ^④
LOGICAL INSTRUCTIONS																	
AND	2	60	3	4	61	4	5	62	3	6/11	3	7/12	63	4	6/11	5	7/12
AND	3	40	4	5	41	5	6	42	4	7/12	4	8/13	43	5	7/12	6	8/13
ANDB	2	70	3	4	71	3	4	72	3	6/11	3	7/12	73	4	6/11	5	7/12
ANDB	3	50	4	5	51	4	5	52	4	7/12	4	8/13	53	5	7/12	6	8/13
OR	2	80	3	4	81	4	5	82	3	6/11	3	7/12	83	4	6/11	5	7/12
ORB	2	90	3	4	91	3	4	92	3	6/11	3	7/12	93	4	6/11	5	7/12
XOR	2	84	3	4	85	4	5	86	3	6/11	3	7/12	87	4	6/11	5	7/12
XORB	2	94	3	4	95	3	4	96	3	6/11	3	7/12	97	4	6/11	5	7/12
DATA TRANSFER INSTRUCTIONS																	
LD	2	A0	3	4	A1	4	5	A2	3	6/11	3	7/12	A3	4	6/11	5	7/12
LDB	2	B0	3	4	B1	3	4	B2	3	6/11	3	7/12	B3	4	6/11	5	7/12
ST	2	C0	3	4	—	—	—	C2	3	7/11	3	8/12	C3	4	7/11	5	8/12
STB	2	C4	3	4	—	—	—	C6	3	7/11	3	8/12	C7	4	7/11	5	8/12
LDBSE	2	BC	3	4	BD	3	4	BE	3	6/11	3	7/12	BF	4	6/11	5	7/12
LDBZE	2	AC	3	4	AD	3	4	AE	3	6/11	3	7/12	AF	4	6/11	5	7/12
STACK OPERATIONS (internal stack)																	
PUSH	1	C8	2	8	C9	3	8	CA	2	11/15	2	12/16	CB	3	11/15	4	12/16
POP	1	CC	2	12	—	—	—	CE	2	14/18	2	14/18	CF	3	14/18	4	14/18
PUSHF	0	F2	1	8													
POPF	0	F3	1	9													
STACK OPERATIONS (external stack)																	
PUSH	1	C8	2	12	C9	3	12	CA	2	15/19	2	16/20	CB	3	15/19	4	16/20
POP	1	CC	2	14	—	—	—	CE	2	16/20	2	16/20	CF	3	16/20	4	16/20
PUSHF	0	F2	1	12													
POPF	0	F3	1	13													
JUMPS AND CALLS																	
MNEMONIC	OPCODE	BYTES	STATES	MNEMONIC	OPCODE	BYTES	STATES										
LJMP	E7	3	8	LCALL	EF	3	13/16 ^⑤										
SJMP	20-27 ^④	2	8	SCALL	28-2F ^④	2	13/16 ^⑤										
BR[]	E3	2	8	RET	F0	1	12/16 ^⑤										
				TRAP ^③	F7	1	21/24										

NOTES:

- ① Number of state times shown for internal/external operands.
- ② The assembler does not accept this mnemonic.
- ③ The least significant 3 bits of the opcode are concatenated with the following 8 bits to form an 11-bit, 2's complement, offset for the relative call or jump.
- ④ State times for stack located internal/external.
- ⑤ State times shown for 16-bit bus.

CONDITIONAL JUMPS

All conditional jumps are 2 byte instructions. They require 8 state times if the jump is taken, 4 if it is not.⁽⁸⁾

MNEMONIC	OPCODE	MNEMONIC	OPCODE	MNEMONIC	OPCODE	MNEMONIC	OPCODE
JC	DB	JE	DF	JGE	D6	JGT	D2
JNC	D3	JNE	D7	JLT	DE	JLE	DA
JH	D9	JV	DD	JVT	DC	JST	D8
JNH	D1	JNV	D5	JNVT	D4	JNST	D0

JUMP ON BIT CLEAR OR BIT SET

These instructions are 3-byte instructions. They require 9 state times if the jump is taken, 5 if it is not.⁽⁸⁾

MNEMONIC	BIT NUMBER							
	0	1	2	3	4	5	6	7
JBC	30	31	32	33	34	35	36	37
JBS	38	39	3A	3B	3C	3D	3E	3F

LOOP CONTROL

MNEMONIC	OPCODE	BYTES	STATE TIMES
DJNZ	EO	3	5/9 STATE TIME (NOT TAKEN/TAKEN) ⁽⁸⁾

SINGLE REGISTER INSTRUCTIONS

MNEMONIC	OPCODE	BYTES	STATES ⁽⁸⁾	MNEMONIC	OPCODE	BYTES	STATES ⁽⁸⁾
DEC	05	2	4	EXT	06	2	4
DECB	15	2	4	EXTB	16	2	4
NEG	03	2	4	NOT	02	2	4
NEGB	13	2	4	NOTB	12	2	4
INC	07	2	4	CLR	01	2	4
INCB	17	2	4	CLRB	11	2	4

SHIFT INSTRUCTIONS

INSTR MNEMONIC	WORD		INSTR MNEMONIC	BYTE		INSTR MNEMONIC	DBL WD		STATE TIMES ⁽⁸⁾
	OP	B		OP	B		OP	B	
SHL	09	3	SHLB	19	3	SHLL	0D	3	7 + 1 PER SHIFT ⁽⁷⁾
SHR	08	3	SHRB	18	3	SHRL	0C	3	7 + 1 PER SHIFT ⁽⁷⁾
SHRA	0A	3	SHRAB	1A	3	SHRAL	0E	3	7 + 1 PER SHIFT ⁽⁷⁾

SPECIAL CONTROL INSTRUCTIONS

MNEMONIC	OPCODE	BYTES	STATES ⁽⁸⁾	MNEMONIC	OPCODE	BYTES	STATES ⁽⁸⁾
SETC	F9	1	4	DI	FA	1	4
CLRC	F8	1	4	EI	FB	1	4
CLRV	FC	1	4	NOP	FD	1	4
RST ⁽⁶⁾	FF	1	166	SKIP	00	2	4

NORMALIZE

MNEMONIC	OPCODE	BYTES	STATE TIMES
NORML	0F	3	11 + 1 PER SHIFT

NOTES:

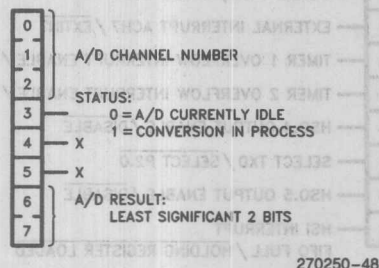
6. This instruction takes 2 states to pull RESET low, then holds it low for 2 states to initiate a reset. The reset takes 12 states, at which time the program restarts at location 2080H. If a capacitor is tied to RESET, the pin may take longer to go low and may never reach the V_{OL} specification.

7. Execution will take at least 8 states, even for 0 shift.

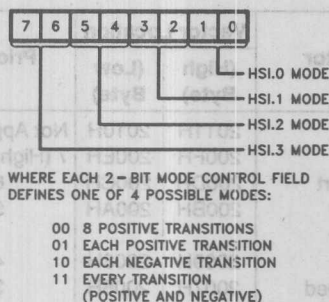
8. State times shown for 16-bit bus.

14.9 SFR Summary

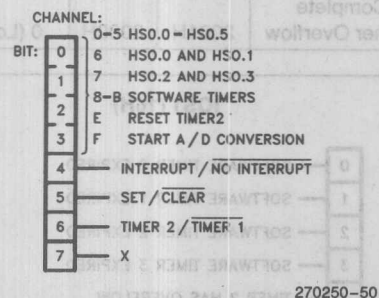
A/D Result LO (02H)



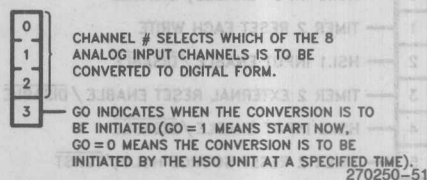
HSI_Mode (03H)



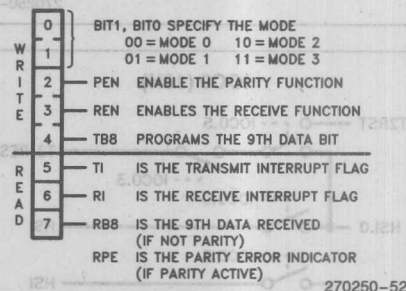
HSO Command (06H)



A/D Command (02H)



SPCON/SPSTAT (11H)



Baud Rate Calculations

Using XTAL1:

$$\text{Mode 0: Baud Rate} = \frac{\text{XTAL1 frequency}}{4 \cdot (B + 1)}; B \neq 0$$

$$\text{Others: Baud Rate} = \frac{\text{XTAL1 frequency}}{64 \cdot (B + 1)}$$

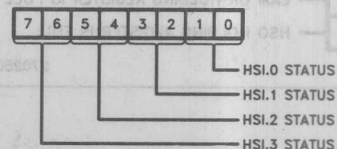
Using T2CLK:

$$\text{Mode 0: Baud Rate} = \frac{\text{T2CLK frequency}}{B}; B \neq 0$$

$$\text{Others: Baud Rate} = \frac{\text{T2CLK frequency}}{16 \cdot B}; B \neq 0$$

Note that B cannot equal 0, except when using XTAL1 in other than Mode 0.

HSI_Status (06H)

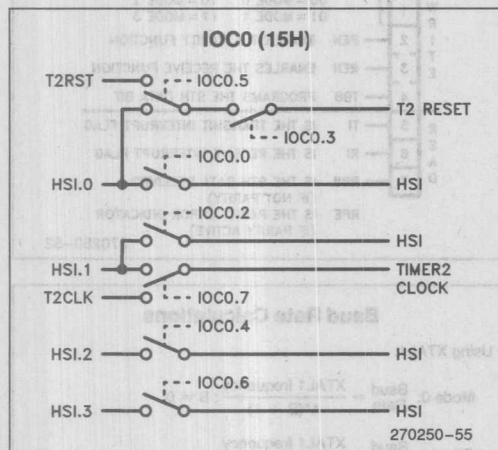


WHERE FOR EACH 2-BIT STATUS FIELD THE LOWER BIT INDICATES WHETHER OR NOT AN EVENT HAS OCCURRED ON THIS PIN AND THE UPPER BIT INDICATES THE CURRENT STATUS OF THE PIN.

270250-53

0	HSI.0 INPUT ENABLE / DISABLE
1	TIMER 2 RESET EACH WRITE
2	HSI.1 INPUT ENABLE / DISABLE
3	TIMER 2 EXTERNAL RESET ENABLE / DISABLE
4	HSI.2 INPUT ENABLE / DISABLE
5	TIMER 2 RESET SOURCE HSI.0 / T2RST
6	HSI.3 INPUT ENABLE / DISABLE
7	TIMER 2 CLOCK SOURCE HSI.1 / T2CLK

270250-54



0	HSO.0 CURRENT STATE
1	HSO.1 CURRENT STATE
2	HSO.2 CURRENT STATE
3	HSO.3 CURRENT STATE
4	HSO.4 CURRENT STATE
5	HSO.5 CURRENT STATE
6	CAM OR HOLDING REGISTER IS FULL
7	HSO HOLDING REGISTER IS FULL

270250-56

0	SELECT PWM / SELECT P2.5
1	EXTERNAL INTERRUPT ACH7 / EXTINT
2	TIMER 1 OVERFLOW INTERRUPT ENABLE / DISABLE
3	TIMER 2 OVERFLOW INTERRUPT ENABLE / DISABLE
4	HSO.4 OUTPUT ENABLE / DISABLE
5	SELECT TXD / SELECT P2.0
6	HSO.5 OUTPUT ENABLE / DISABLE
7	HSI INTERRUPT FIFO FULL / HOLDING REGISTER LOADED

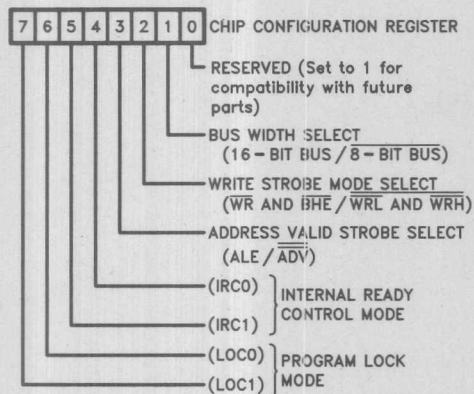
270250-57

Vector	Vector Location		Priority
	(High Byte)	(Low Byte)	
Software	2011H	2010H	Not Applicable
Extint	200FH	200EH	7 (Highest)
Serial Port	200DH	200CH	6
Software Timers	200BH	200AH	5
HSI.0	2009H	2008H	4
High Speed Outputs	2007H	2006H	3
HSI Data Available	2005H	2004H	2
A/D Conversion Complete	2003H	2002H	1
Timer Overflow	2001H	2000H	0 (Lowest)

0	SOFTWARE TIMER 0 EXPIRED
1	SOFTWARE TIMER 1 EXPIRED
2	SOFTWARE TIMER 2 EXPIRED
3	SOFTWARE TIMER 3 EXPIRED
4	TIMER 2 HAS OVERFLOW
5	TIMER 1 HAS OVERFLOW
6	HSI FIFO IS FULL
7	HSI HOLDING REGISTER DATA AVAILABLE

270250-58

Chip Configuration



270250-59

Internal Ready Control

IRC1	IRC0	Description
0	0	Limit to 1 Wait State
0	1	Limit to 2 Wait States
1	0	Limit to 3 Wait States
1	1	Disable Internal Ready Control

Program Lock Modes

LOC1	LOC0	Protection
0	0	Read and Write Protected
0	1	Read Protected
1	0	Write Protected
1	1	No Protection

Programming Function PMODE Values

PMODE	Programming Mode
0-4	Reserved
5	Slave Programming
6-0BH	Reserved
0CH	Auto Programming Mode
0DH	Program Configuration Byte
0EH-0FH	Reserved

Slave Programming Mode Commands

P4.7	P4.6	Action
0	0	Word Dump
0	1	Data Verify
1	0	Data Program
1	1	Reserved

8X9XBH Signature Word

Device	Signature Word
879XBH	896FH
839XBH	896EH
809XBH	Undefined

Port 2 Pin Functions

Port	Function	Alternate Function
P2.0	Output	TXD (Serial Port Transmit)
P2.1	Input	RXD (Serial Port Receive)
P2.2	Input	EXTINT (External Interrupt)
P2.3	Input	T2CLK (Timer 2 Clock)
P2.4	Input	T2RST (Timer 2 Reset)
P2.5	Output	PWM (Pulse Width Modulation)

16

MCS®-9d Instruction Set

MCS®-96 INSTRUCTION SET

OVERVIEW

This chapter of the manual gives a description of each instruction recognized by the 8096. The instructions are sorted alphabetically by the mnemonic used in the assembly language for the 8096. A summary of the instruction set is included in Section 14 of the MCS®-96 Architecture chapter.

The instruction set descriptions in the following sections do not always show the effect on the program counter (PC). Unless otherwise specified, all instructions increment the PC by the number of bytes in the instruction.

A set of acronyms are used to make the instruction set descriptions easier to read, their definitions are listed below:

aa. A two bit field within an opcode which selects the basic addressing mode user. This field is only present in those opcodes which allow address mode options. The encoding of the field is as follows:

aa	Addressing mode
00	Register direct
01	Immediate
10	Indirect
11	Indexed

The selection between indirect and indirect with auto-increment or between short and long indexing is done based on the least significant bit of the instruction byte which follows the opcode. This type selects the 16-bit register which is to take part in the address calculation. Since the 8096 requires that words be aligned on even byte boundaries this bit would be otherwise unused.

breg. A byte register in the internal register file. When confusion could exist as to whether this field refers to a source or a destination register it will be prefixed with an "S" or a "D".

baop. A byte operand which is addressed by any of the address modes discussed in Section 3.2 of the MCS-96 Architecture chapter.

bitno. A three bit field within an instruction op-code which selects one of the eight bits in a byte.

wreg. A word register in the internal register file. When confusion could exist as to whether this field refers to a source register or a destination register it will be prefixed with an "S" or a "D".

waop. A word operand which is addressed by any of the address modes discussed in Section 3.2 of the MCS-96 Architecture chapter.

Lreg. A 32-bit register in the internal register file.

BEA. Extra bytes of code required for the address mode selected.

CEA. Extra state times (cycles) required for the address mode selected.

cadd. An address in the program code.

Flag Settings. The modification to the flag setting is shown for each instruction. A checkmark (✓) means that the flag is set or cleared as appropriate. A hyphen means that the flag is not modified. A one or zero (1) or (0) indicates that the flag will be in that state after the instruction. An up arrow (↑) indicates that the instruction may set the flag if it is appropriate but will not clear the flag. A down arrow (↓) indicates that the flag can be cleared but not set by the instruction. A question mark (?) indicates that the flag will be left in an indeterminant state after the operation.

Generic Jumps and Calls. The assembler for the MCS-96 family provides for generic jumps and calls. For all of the conditional jump instructions a "B" can be substituted for the "J" and the assembler will generate a code sequence which is logically equivalent but can reach anywhere in the memory. A JH can only jump about 128 locations from the current program counter; a BH can jump anywhere in memory. In a like manner a BR will cause a SJMP or LJMP to be generated as appropriate and a CALL will cause a SCALL or LCALL to be generated. The assembler user's guide should be consulted for the algorithms used by the assembler to convert these generic instructions into actual machine instructions.

Indirect Shifts. The indirect shift operations use registers 24 through 255 (18H-0FFH), since 0-15 are direct operators and registers 16 through 23 are Special Function Registers. Note that indirect shifts through SFRs are illegal operations.

The maximum shift count is 31 (1FH). Count values above this will be truncated to the 5 least significant bits.

1. ADD (Two Operands) — ADD WORDS

Operation: The sum of the two word operands is stored into the destination (leftmost) operand.

$$(DEST) \leftarrow (DEST) + (SRC)$$

Assembly Language Format:

	DST	SRC
ADD	wreg,	waop

Object Code Format: [011001aa] [waop] [wreg]

Bytes: 2 + BEA

States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

2. ADD (Three Operands) — ADD WORDS

Operation: The Sum of the second and third word operands is stored into the destination (leftmost) operand.

$$(DEST) \leftarrow (SRC1) + (SRC2)$$

Assembly Language Format:

	DST	SRC1	SRC2
ADD	Dwreg, Swreg,	waop	

Object Code Format: [010001aa] [waop] [Swreg] [Dwreg]

Bytes: 3 + BEA

States: 5 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

3. ADDB (Two Operands) — ADD BYTES

Operation: The sum of the two byte operands is stored into the destination (leftmost) operand.

$$(DEST) \leftarrow (DEST) + (SRC)$$

Assembly Language Format:

ADDB DST SRC
breg, baop

Object Code Format: [011101aa] [baop] [breg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

4. ADDB (Three Operands) — ADD BYTES

Operation: The sum of the second and third byte operands is stored into the destination (leftmost) operand.

$$(DEST) \leftarrow (SRC1) + (SRC2)$$

Assembly Language Format:

ADDB DST SRC1 SRC2
Dbreg, Sbreg, baop

Object Code Format: [010101aa] [baop] [Sbreg] [Dbreg]

Bytes: 3 + BEA
States: 5 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

Operation: The sum of the two word operands and the carry flag (0 or 1) is stored into the destination (leftmost) operand.

$$(DEST) \leftarrow (DEST) + (SRC) + C$$

Assembly Language Format:

ADDC DST SRC
 wreg, waop

Object Code Format: [101001aa] [waop] [wreg]

Bytes: 2 + BEA
States: 4 + BEA

Flags Affected					
Z	N	C	V	VT	ST
↓	↗	↗	↗	↑	—

6. ADDCB — ADD BYTES WITH CARRY

Operation: The sum of the two byte operands and the carry flag (0 or 1) is stored into the destination (leftmost) operand.

$$(DEST) \leftarrow (DEST) + (SRC) + C$$

Assembly Language Format:

ADDCB DST SRC
 breg, baop

Object Code Format: [101101aa] [baop] [breg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
↓	↗	↗	↗	↑	—

7. AND (Two Operands) — LOGICAL AND WORDS

Operation: The two word operands are ANDed, the result having a 1 only in those bit positions where both operands had a 1, with zeroes in all other bit positions. The result is stored into the destination (leftmost) operand.

(DEST) ← (DEST) AND (SRC)

Assembly Language Format:

AND DST SRC
wreg, waop

Object Code Format: [011000aa] [waop] [wreg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

8. AND (Three Operands) — LOGICAL AND WORDS

Operation: The second and third word operands are ANDed, the result having a 1 only in those bit positions where both operands had a 1, with zeroes in all other bit positions. The result is stored into the destination (leftmost) operand.

(DEST) ← (SRC1) AND (SRC2)

Assembly Language Format:

AND DST SRC1 SRC2
Dwreg, Swreg, waop

Object Code Format: [010000aa] [waop] [Swreg] [Dwreg]

Bytes: 3 + BEA
States: 5 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

9. ANDB (Two Operands) — LOGICAL AND BYTES

Operation: The two byte operands are ANDed, the result having a 1 only in those bit positions where both operands had a 1, with zeroes in all other bit positions. The result is stored into the destination (leftmost) operand.

(DEST) ← (DEST) AND (SRC)

Assembly Language Format:

ANDB DST SRC
breg, baop

Object Code Format: [011100aa] [baop] [breg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

10. ANDB (Three Operands) — LOGICAL AND BYTES

Operation: The second and third byte operands are ANDed, the result having a 1 only in those bit positions where both operands had a 1, with zeroes in all other bit positions. The result is stored into the destination (leftmost) operand.

(DEST) ← (SRC1) AND (SRC2)

Assembly Language Format:

ANDB DST SRC1 SRC2
Dbreg, Sbreg, baop

Object Code Format: [010100aa] [baop] [Sbreg] [Dbreg]

Bytes: 3 + BEA
States: 5 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

11. BR (Indirect) — BRANCH INDIRECT

Operation: The execution continues at the address specified in the operand word register.

$PC \leftarrow (DEST)$

Assembly Language Format: BR [wreg]

Object Code Format: [11100011] [wreg]

Bytes: 2
States: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

12. CLR — CLEAR WORD

Operation: The value of the word operand is set to zero.

$(DEST) \leftarrow 0$

Assembly Language Format: CLR wreg

Object Code Format: [00000001] [wreg]

Bytes: 2
States: 4

Flags Affected					
Z	N	C	V	VT	ST
1	0	0	0	—	—

Operation: The value of the byte operand is set to zero.

(DEST) ← 0

Assembly Language Format: CLRB breg

Object Code Format: [00010001] [breg]

Bytes: 2

States: 4

Flags Affected					
Z	N	C	V	VT	ST
1	0	0	0	—	—

14. CLRC — CLEAR CARRY FLAG

Operation: The value of the carry flag is set to zero.

C ← 0

Assembly Language Format: CLRC

Object Code Format: [11111000]

Bytes: 1

States: 4

Flags Affected					
Z	N	C	V	VT	ST
—	—	0	—	—	—

15. CLRVT — CLEAR OVERFLOW TRAP

Operation: The value of the overflow-trap flag is set to zero.
 $VT \leftarrow 0$

Assembly Language Format: CLRVT

Object Code Format: [11111100]

Bytes: 1
 States: 4

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	0	—

16. CMP — COMPARE WORDS

Operation: The source (rightmost) word operand is subtracted from the destination (leftmost) word operand. The flags are altered but the operands remain unaffected. The carry flag is set as complement of borrow.

(DEST) — (SRC)

Assembly Language Format: DST SRC
 CMP wreg, waop

Object Code Format: [100010aa] [waop] [wreg]

Bytes: 2 + BEA
 States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

17. CMPB — COMPARE BYTES

Operation: The source (rightmost) byte operand is subtracted from the destination (leftmost) byte operand. The flags are altered but the operands remain unaffected. The carry flag is set as complement of borrow.
(DEST) ← (DEST) - (SRC)

Assembly Language Format:

CMPB DST SRC
breg, baop

Object Code Format: [100110aa] [baop] [breg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

18. DEC — DECREMENT WORD

Operation: The value of the word operand is decremented by one.
(DEST) ← (DEST) - 1

Assembly Language Format:

DEC wreg

Object Code Format: [00000101] [wreg]

Bytes: 2
States: 4

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

19. DECB — DECREMENT BYTE

Operation: The value of the byte operand is decremented by one.
 $(DEST) \leftarrow (DEST) - 1$

Assembly Language Format: DECB breg

Object Code Format: [00010101] [breg]

Bytes: 2
 States: 4

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

20. DI — DISABLE INTERRUPTS

Operation: Interrupts are disabled. Interrupt-calls will not occur after this instruction.
 Interrupt Enable (PSW.9) $\leftarrow 0$

Assembly Language Format: DI

Object Code Format: [11111010]

Bytes: 1
 States: 4

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

Operation: This instruction divides the contents of the destination LONG-INTEGGER operand by the contents of the INTEGER word operand, using signed arithmetic. The low order word of the destination (i.e., the word with the lower address) will contain the quotient; the high order word will contain the remainder.

(low word DEST) \leftarrow (DEST) / (SRC)

(high word DEST) \leftarrow (DEST) MOD (SRC)

The above two statements are performed concurrently.

Assembly Language Format:

DST SRC
DIV lreg, waop

Object Code Format: [11111110] [100011aa] [waop] [lreg]

Bytes: 2 + BEA
States: 29 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	?	↑	—

22. DIVB — DIVIDE SHORT-INTEGERS

Operation: This instruction divides the contents of the destination INTEGER operand by the contents of the source SHORT-INTEGGER operand, using signed arithmetic. The low order byte of the destination (i.e., the byte with the lower address) will contain the quotient; the high order byte will contain the remainder.

(low byte DEST) \leftarrow (DEST) / (SRC)

(high byte DEST) \leftarrow (DEST) MOD (SRC)

The above two statements are performed concurrently.

Assembly Language Format:

DST SRC
DIVB wreg, baop

Object Code Format: [11111110] [100111aa] [baop] [wreg]

Bytes: 2 + BEA
States: 21 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	?	↑	—

23. DIVU — DIVIDE WORDS

Operation: This instruction divides the content of the destination DOUBLE-WORD operand by the contents of the source WORD operand, using unsigned arithmetic. The low order word will contain the quotient; the high order WORD will contain the remainder.

(low word DEST) \leftarrow (DEST) / (SRC)

(high word DEST) \leftarrow (DEST) MOD (SRC)

The above two statements are performed concurrently.

Assembly Language Format:

DST SRC
DIVU lreg, waop

Object Code Format: [100011aa] [waop] [lreg]

Bytes: 2 + BEA

States: 25 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	✓	↑	—

24. DIVUB — DIVIDE BYTES

Operation: This instruction divides the contents of the destination WORD operand by the contents of the source BYTE operand, using unsigned arithmetic. The low order byte of the destination, (i.e., the byte with the lower address) will contain the quotient; the high order byte will contain the remainder.

(low byte DEST) \leftarrow (DEST) / (SRC)

(high byte DEST) \leftarrow (DEST) MOD (SRC)

The above two statements are performed concurrently.

Assembly Language Format:

DST SRC
DIVUB wreg, baop

Object Code Format: [100111aa] [baop] [wreg]

Bytes: 2 + BEA

States: 17 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	✓	↑	—

25. DJNZ — DECREMENT AND JUMP IF NOT ZERO

Operation: The value of the byte operand is decremented by 1. If the result is not equal to 0, the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the result of the decrement is zero then control passes to the next sequential instruction.

(COUNT) ← (COUNT) - 1
if (COUNT) <> 0 then
PC ← PC + disp (sign-extended to 16 bits)
end_if

Assembly Language Format: DJNZ breg, cadd

Object Code Format: [11100000] [breg] [disp]

Bytes: 3
States: Jump Not Taken: 5
Jump Taken: 9

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

26. EI — ENABLE INTERRUPTS

Operation: Interrupts are enabled following the execution of the next statement. Interrupt-calls cannot occur immediately following this instruction.

Interrupt Enable (PSW.9) ← 1

Assembly Language Format: EI

Object Code Format: [11111011]

Bytes: 1
States: 4

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

27. EXT — SIGN EXTEND INTEGER INTO LONG-INTEGER

Operation: The low order word of the operand is sign-extended throughout the high order word of the operand.

if (low word DEST) < 8000H then

(high word DEST) ← 0

else

(high word DEST) ← 0FFFFH

end_if

Assembly Language Format: EXT lreg

Object Code Format: [00000110] [lreg]

Bytes: 2

States: 4

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

28. EXTB — SIGN EXTEND SHORT-INTEGER INTO INTEGER

Operation: The low order byte of the operand is sign-extended throughout the high order byte of the operand.

if (low byte DEST) < 80H then

(high byte DEST) ← 0

else

(high byte DEST) ← 0FFH

end_if

Assembly Language Format: EXTB wreg

Object Code Format: [00010110] [wreg]

Bytes: 2

States: 4

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

27. EXT — SIGN EXTEND INTO LONG-WORD

Operation: The value of the word operand is incremented by 1.

$$(DEST) \leftarrow (DEST) + 1$$

Assembly Language Format: INC wreg

Object Code Format: [00000111] [wreg]

Bytes: 2

States: 4

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

30. INCB — INCREMENT BYTE

Operation: The value of the byte operand is incremented by 1.

$$(DEST) \leftarrow (DEST) + 1$$

Assembly Language Format: INCB breg

Object Code Format: [00010111] [breg]

Bytes: 2

States: 4

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

31. JBC — JUMP IF BIT CLEAR

Operation: The specified bit is tested. If it is clear (i.e., 0), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the bit is set (i.e., 1), control passes to the next sequential instruction.

if (specified bit) = 0 then
PC ← PC + disp (sign-extended to 16 bits)

Assembly Language Format: JBC breg,bitno,cadd

Object Code Format: [00110bbb] [breg] [disp]
where bbb is the bit number within the specified register.

Bytes: 3
States: Jump Not Taken: 5
Jump Taken: 9

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

33. JC — JUMP IF CARRY FLAG IS SET

Operation: If the carry flag is set (i.e., 1), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the carry flag is clear (i.e., 0), control passes to the next sequential instruction.

if C = 1 then
PC ← PC + disp (sign-extended to 16 bits)

Assembly Language Format: JC cadd

Object Code Format: [11011011] [disp]

Bytes: 2
States: Jump Not Taken: 4
Jump Taken: 6

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

32. JBS — JUMP IF BIT SET

Operation: The specified bit is tested. If it is set (i.e., 1), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to $+127$. If the bit is clear (i.e., 0), control passes to the next sequential instruction.

if (specified bit) = 1 then
 $PC \leftarrow PC + \text{disp}$ (sign-extended to 16 bits)

Assembly Language Format: JBS breg,bitno,cadd

Object Code Format: [00111bbb] [breg] [disp]
 where bbb is the bit number within the specified register.

Bytes: 3
 States: Jump Not Taken: 5
 Jump Taken: 9

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

33. JC — JUMP IF CARRY FLAG IS SET

Operation: If the carry flag is set (i.e., 1), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to $+127$. If the carry flag is clear (i.e., 0), control passes to the next sequential instruction.

if C = 1 then
 $PC \leftarrow PC + \text{disp}$ (sign-extended to 16 bits)

Assembly Language Format: JC cadd

Object Code Format: [11011011] [disp]

Bytes: 2
 States: Jump Not Taken: 4
 Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

34. JE — JUMP IF EQUAL

Operation: If the zero flag is set (i.e., 1), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the zero flag is clear (i.e., 0), control passes to the next sequential instruction.

if Z = 1 then
 $PC \leftarrow PC + \text{disp (sign-extended to 16 bits)}$

Assembly Language Format: JE cadd

Object Code Format: [11011111] [disp]

Bytes: 2
 States: Jump Not Taken: 4
 Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

35. JGE — JUMP IF SIGNED GREATER THAN OR EQUAL

Operation: If the negative flag is clear (i.e., 0), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the negative flag is set (i.e., 1), control passes to the next sequential instruction.

if N = 0 then
 $PC \leftarrow PC + \text{disp (sign-extended to 16 bits)}$

Assembly Language Format: JGE cadd

Object Code Format: [11010110] [disp]

Bytes: 2
 States: Jump Not Taken: 4
 Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

Operation: If both the negative flag and the zero flag are clear (i.e., 0), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If either the negative flag or the zero flag are set (i.e., 1,) control passes to the next sequential instruction.

if $N = 0$ AND $Z = 0$ then

$PC \leftarrow PC + \text{disp (sign-extended to 16 bits)}$

Assembly Language Format: JGT cadd

Object Code Format: [11010010] [[disp]]

Bytes: 2
States: Jump Not Taken: 4
Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

37. JH — JUMP IF HIGHER (UNSIGNED)

Operation: If the carry flag is set (i.e., 1), but the zero flag is not, the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If either the carry flag is clear or the zero flag is set, control passes to the next sequential instruction.

if $C = 1$ AND $Z = 0$ then

$PC \leftarrow PC + \text{disp (sign-extended to 16 bits)}$

Assembly Language Format: JH cadd

Object Code Format: [11011001] [[disp]]

Bytes: 2
States: Jump Not Taken: 4
Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

38. JLE — JUMP IF SIGNED LESS THAN OR EQUAL

Operation: If either the negative flag or the zero flag are set (i.e., 1), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to $+127$. If both the negative flag and the zero flag are clear (i.e., 0), control passes to the next sequential instruction.

if $N = 1$ OR $Z = 1$ then

$PC \leftarrow PC + \text{disp}$ (sign-extended to 16 bits)

Assembly Language Format: JLE cadd

Object Code Format: [11011010] [disp]

Bytes: 2
States: Jump Not Taken: 4
Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

39. JLT — JUMP IF SIGNED LESS THAN

Operation: If the negative flag is set (i.e., 1), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to $+127$. If the negative flag is clear (i.e., 0), control passes to the next sequential instruction.

if $N = 1$ then

$PC \leftarrow PC + \text{disp}$ (sign-extended to 16 bits)

Assembly Language Format: JLT cadd

Object Code Format: [11011110] [disp]

Bytes: 2
States: Jump Not Taken: 4
Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

40. JNC — JUMP IF CARRY FLAG IS CLEAR

Operation: If the carry flag is clear (i.e., 0), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the carry flag is set (i.e., 1), control passes to the next sequential instruction.

if C = 0 then

$PC \leftarrow PC + \text{disp (sign-extended to 16 bits)}$

Assembly Language Format: JNC cadd

Object Code Format: [11010011] [disp]

Bytes: 2
States: Jump Not Taken: 4
Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

41. JNE — JUMP IF NOT EQUAL

Operation: If the zero flag is clear (i.e., 0), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the zero flag is set (i.e., 1), control passes to the next sequential instruction.

if Z = 0 then

$PC \leftarrow PC + \text{disp (sign-extended to 16 bits)}$

Assembly Language Format: JNE cadd

Object Code Format: [11010111] [disp]

Bytes: 2
States: Jump Not Taken: 4
Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

42. JNH — JUMP IF NOT HIGHER (UNSIGNED)

Operation: If either the carry flag is clear (i.e., 0), or the zero flag is set (i.e., 1), the distance from the end of this instruction to the target label is added to program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the carry flag is set (i.e., 1) and the zero flag is not, control passes to the next sequential instruction.

if $C = 0$ OR $Z = 1$ then
 $PC \leftarrow PC + \text{disp}$ (sign-extended to 16 bits)

Assembly Language Format: JNH cadd

Object Code Format: [11010001] [disp]

Bytes: 2
 States: Jump Not Taken: 4
 Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

43. JNST — JUMP IF STICKY BIT IS CLEAR

Operation: If the sticky bit flag is clear (i.e., 0), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the sticky bit flag is set (i.e., 1), control passes to the next sequential instruction.

if $ST = 0$ then
 $PC \leftarrow PC + \text{disp}$ (sign-extended to 16 bits)

Assembly Language Format: JNST cadd

Object Code Format: [11010000] [disp]

Bytes: 2
 States: Jump Not Taken: 4
 Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

Operation: If the overflow flag is clear (i.e., 0), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the overflow flag is set (i.e., 1), control passes to next sequential instruction.

if V = 0 then
 $PC \leftarrow PC + \text{disp (sign-extended to 16 bits)}$

Assembly Language Format: JNV cadd

Object Code Format: [11010101] [] [disp]

Bytes: 2
 States: Jump Not Taken: 4
 Jump Taken: 8

Flags Affected							
Z	N	C	V	VT	ST		
—	—	—	—	—	—	—	—

45. JNVT — JUMP IF OVERFLOW TRAP IS CLEAR

Operation: If the overflow trap flag is clear (i.e., 0), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the overflow trap flag is set (i.e., 1), control passes to the next sequential instruction. The VT flag is cleared.

if VT = 0 then
 $PC \leftarrow PC + \text{disp (sign-extended to 16 bits)}$

Assembly Language Format: JNVT cadd

Object Code Format: [11010100] [] [disp]

Bytes: 2
 States: Jump Not Taken: 4
 Jumps Taken: 8

Flags Affected							
Z	N	C	V	VT	ST		
—	—	—	—	0	—	—	—

46. JST — JUMP IF STICKY BIT IS SET

Operation: If the sticky bit flag is set (i.e., 1), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the sticky bit flag is clear (i.e., 0), control passes to the next sequential instruction.

if ST = 1 then
 $PC \leftarrow PC + \text{disp}$ (sign-extended to 16 bits)

Assembly Language Format: JST cadd

Object Code Format: [11011000] [[disp]]

Bytes: 2
 States: Jump Not Taken: 4
 Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

47. JV — JUMP IF OVERFLOW FLAG IS SET

Operation: If the overflow is set (i.e., 1), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the overflow flag is clear (i.e., 0), control passes to the next sequential instruction.

if V = 1 then
 $PC \leftarrow PC + \text{disp}$ (sign-extended to 16 bits)

Assembly Language Format: JV cadd

Object Code Format: [11011101] [[disp]]

Bytes: 2
 States: Jump Not Taken: 4
 Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

48. JVT — JUMP IF OVERFLOW TRAP IS SET

Operation: If the overflow trap flag is set (i.e., 1), the distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the target label must be in the range of -128 to +127. If the overflow trap flag is clear (i.e., 0), control passes to the next sequential instruction. The VT flag is cleared.

if VT = 1 then

PC ← PC + disp (sign-extended to 16 bits)

Assembly Language Format: JVT cadd

Object Code Format: [11011100] [disp]

Bytes: 2
States: Jump Not Taken: 4
Jump Taken: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	0	—

49. LCALL — LONG CALL

Operation: The contents of the program counter (the return address) is pushed onto the stack. Then the distance from the end of this instruction to the target label is added to the program counter, effecting the call. The operand may be any address in the entire address space.

SP ← SP - 2

(SP) ← PC

PC ← PC + disp

Assembly Language Format: LCALL cadd

Object Code Format: [11101111] [disp-low] [disp-hi]

Bytes: 3
States: Onchip stack: 13
Offchip stack: 16

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

50. LD — LOAD WORD

Operation: The value of the source (rightmost) word operand is stored into the destination (leftmost) operand.

(DEST) ← (SRC)

Assembly Language Format:

LD DST SRC
wreg, waop

Object Code Format: [101000aa] [waop] [wreg]

Bytes: 2 + BEA

States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

51. LDB — LOAD BYTE

Operation: The value of the source (rightmost) byte operand is stored into the destination (leftmost) operand.

(DEST) ← (SRC)

Assembly Language Format:

LDB DST SRC
breg, baop

Object Code Format: [101100aa] [baop] [breg]

Bytes: 2 + BEA

States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

52. LDBSE — LOAD INTEGER WITH SHORT-INTEGER

Operation: The value of the source (rightmost) byte operand is sign-extended and stored into the destination (leftmost) word operand.

(low byte DEST) \leftarrow (SRC)
 if (SRC) < 80H then
 (high byte DEST) \leftarrow 0
 else
 (high byte DEST) \leftarrow 0FFH
 end_if

Assembly Language Format:

LDBSE DST SRC
 wreg, baop

Object Code Format: [101111aa] [baop] [wreg]

Bytes: 2 + BEA
 States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

53. LDBZE — LOAD WORD WITH BYTE

Operation: The value of the source (rightmost) byte operand is zero-extended and stored into the destination (leftmost) word operand.

(low byte DEST) \leftarrow (SRC)
 (high byte DEST) \leftarrow 0

Assembly Language Format:

LDBZE DST SRC
 wreg, baop

Object Code Format: [101011aa] [baop] [wreg]

Bytes: 2 + BEA
 States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

Operation: The distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The operand may be any address in the entire address space.

$PC \leftarrow PC + \text{disp}$

Assembly Language Format: LJMP cadd

Object Code Format: [11100111] [disp-low] [disp-hi]

Bytes: 3

States: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

55. MUL (Two Operands) — MULTIPLY INTEGERS

Operation: The two INTEGER operands are multiplied using signed arithmetic and the 32-bit result is stored into the destination (leftmost) LONG-INTEGGER operand. The sticky bit flag is undefined after the instruction is executed.

$(\text{DEST}) \leftarrow (\text{DEST}) * (\text{SRC})$

Assembly Language Format:

MUL DST SRC
lreg, waop

Object Code Format: [11111110] [01011aa] [waop] [lreg]

Bytes: 3 + BEA

States: 29 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	?

56. MUL (Three Operands) — MULTIPLY INTEGERS

Operation: The second and third INTEGER operands are multiplied using signed arithmetic and the 32-bit result is stored into the destination (leftmost) LONG INTEGER operand. The sticky bit flag is undefined after the instruction is executed.

$$(DEST) \leftarrow (SRC1) * (SRC2)$$

Assembly Language Format:

MUL DST SRC1 SRC2
 lreg, wreg, waop

Object Code Format: [11111110] [010011aa] [waop] [lreg]

Bytes: 4 + BEA
States: 30 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	?

57. MULB (Two Operands) — MULTIPLY SHORT-INTEGERS

Operation: The two SHORT-INTEGER operands are multiplied using signed arithmetic and the 16-bit result is stored into the destination (leftmost) INTEGER operand. The sticky bit flag is undefined after the instruction is executed.

$$(DEST) \leftarrow (DEST) * (SRC)$$

Assembly Language Format:

MULB DST SRC
 lreg, baop

Object Code Format: [11111110] [011111aa] [baop] [lreg]

Bytes: 3 + BEA
States: 21 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	?

58. MULB (Three Operands) — MULTIPLY SHORT-INTEGERS

Operation: The second and third SHORT-INTEGER operands are multiplied using signed arithmetic and the 16-bit result is stored into the destination (leftmost) INTEGER operand. The sticky bit flag is undefined after the instruction is executed.

$$(DEST) \leftarrow (SRC1) * (SRC2)$$

Assembly Language Format:

	DST	SRC1	SRC2
MULB	wreg,	breg	baop

Object Code Format: [11111110] [010111aa] [baop] [breg] [wreg]

Bytes: 4 + BEA
States: 22 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	?

59. MULU (Two Operands) — MULTIPLY WORDS

Operation: The two WORD operands are multiplied using unsigned arithmetic and the 32-bit result is stored into the destination (leftmost) DOUBLE-WORD operand. The sticky bit flag is undefined after the instruction is executed.

$$(DEST) \leftarrow (DEST) * (SRC)$$

Assembly Language Format:

	DST	SRC
MULU	lreg,	waop

Object Code Format: [011011aa] [waop] [lreg]

Bytes: 2 + BEA
States: 25 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	?

Operation: The second and third WORD operands are multiplied using unsigned arithmetic and the 32-bit result is stored into the destination (leftmost) DOUBLE-WORD operand. The sticky bit flag is undefined after the instruction is executed.

$(DEST) \leftarrow (SRC1) * (SRC2)$

Assembly Language Format:

MULU DST SRC1 SRC2
 lreg, wreg, waop

Object Code Format: [010011aa] [waop] [lreg]

Bytes: 3 + BEA

States: 26 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	?

61. MULUB (Two Operands) — MULTIPLY BYTES

Operation: The two BYTE operands are multiplied using unsigned arithmetic and the WORD result is stored into the destination (leftmost) operand. The sticky bit flag is undefined after the instruction is executed.

$(DEST) \leftarrow (DEST) * (SRC)$

Assembly Language Format:

MULUB DST SRC
 wreg, baop

Object Code Format: [011111aa] [baop] [wreg]

Bytes: 2 + BEA

States: 17 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	?

62. MULUB (Three Operands) — MULTIPLY BYTES

Operation: The second and third BYTE operands are multiplied using unsigned arithmetic and the WORD result is stored into the destination (leftmost) operand. The sticky bit flag is undefined after the instruction is executed.

$$(DEST) \leftarrow (SRC1) * (SRC2)$$

Assembly Language Format:

MULUB DST SRC1 SRC2
 wreg, breg, baop

Object Code Format: [010111aa] [baop] [breg] [wreg]

Bytes: 3 + BEA
States: 18 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	?

63. NEG — NEGATE INTEGER

Operation: The value of the INTEGER operand is negated.

$$(DEST) \leftarrow -(DEST)$$

Assembly Language Format: NEG wreg

Object Code Format: [00000011] [wreg]

Bytes: 2
States: 4

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

64. NEGB — NEGATE SHORT-INTEGER

Operation: The value of the SHORT-INTEGER operand is negated.
 $(DEST) \leftarrow -(DEST)$

Assembly Language Format: NEGB breg

Object Code Format: [00010011] [breg]

Bytes: 2
 States: 4

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

65. NOP — NO OPERATION

Operation: Nothing is done. Control passes to the next sequential instruction.

Assembly Language Format: NOP

Object Code Format: [11111101]

Bytes: 1
 States: 4

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

66. NORML — NORMALIZE LONG-INTEGERS

Operation: The LONG-INTEGERS operand is normalized; i.e., it is shifted to the left until its most significant bit is 1. If the most significant bit is still 0 after 31 shifts, the process stops and the zero flag is set. The number of shifts actually performed is stored in the second operand.

```
(COUNT) ← 0
do while (MSB(DEST) = 0) AND ((COUNT) < 31)
    (DEST) ← (DEST) * 2
    (COUNT) ← (COUNT) + 1
end_while
```

Assembly Language Format: NORML lreg,breg

Object Code Format: [00001111] [breg] [lreg]

Bytes: 3
States: 11 + No. of shifts performed

Flags Affected					
Z	N	C	V	VT	ST
✓	?	0	—	—	—

67. NOT — COMPLEMENT WORD

Operation: The value of the WORD operand is complemented: each 1 is replaced with a 0, and each 0 with a 1.

```
(DEST) ← NOT (DEST)
```

Assembly Language Format: NOT wreg

Object Code Format: [00000010] [wreg]

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

Operation: The value of the BYTE operand is complemented: each 1 is replaced with a 0, and each 0 with a 1.
 $(DEST) \leftarrow NOT (DEST)$

Assembly Language Format: NOTB breg

Object Code Format: [00010010] [breg]

Bytes: 2
 States: 4

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

69. OR — LOGICAL OR WORDS

Operation: The source (rightmost) WORD is ORed with the destination (leftmost) WORD operand. Each bit is set to 1 if the corresponding bit in either the source operand or the destination operand is 1. The result replaces the original destination operand.

$(DEST) \leftarrow (DEST) OR (SRC)$

Assembly Language Format: OR DST SRC
 OR wreg, waop

Object Code Format: [100000aa] [waop] [wreg]

Bytes: 2 + BEA
 States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

70. ORB — LOGICAL OR BYTES

Operation: The source (rightmost) BYTE operand is ORed with the destination (leftmost) BYTE operand. Each bit is set to 1 if the corresponding bit in either the source operand or the destination operand was 1. The result replaces the original destination operand.

(DEST) ← (DEST) OR (SRC)

Assembly Language Format: ORB breg,baop

Object Code Format: [100100aa] [baop] [breg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

71. POP — POP WORD

Operation: The word on top of the stack is popped and placed at the destination operand.

(DEST) ← (SP)
SP ← SP + 2

Assembly Language Format: POP waop

Object Code Format: [110011aa] [waop]

Bytes: 1 + BEA
States: Onchip Stack: 12 + CEA
Offchip Stack: 14 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

72. POPF — POP FLAGS

Operation: The word on top of the stack is popped and placed in the PSW. Interrupt calls cannot occur immediately following this instruction.

(PSW) ← (SP)
SP ← SP + 2

Assembly Language Format: POPF

Object Code Format: [11110011]

Bytes: 1
States: Onchip Stack: 9
Offchip Stack: 13

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	✓	✓

73. PUSH — PUSH WORD

Operation: The specified operand is pushed onto the stack.

SP ← SP - 2
(SP) ← (DEST)

Assembly Language Format: PUSH waop

Object Code Format: [110010aa] [waop]

Bytes: 1 + BEA
States: Onchip Stack: 8 + CEA
Offchip Stack: 12 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

74. PUSHF — PUSH FLAGS

Operation: The PSW is pushed on top of the stack, and then set to all zeroes. This implies that all interrupts are disabled. Interrupt-calls cannot occur immediately following this instruction.

$SP \leftarrow SP - 2$
 $(SP) \leftarrow PSW$
 $PSW \leftarrow 0$

Assembly Language Format: PUSHF

Object Code Format: [11110010]

Bytes: 1
 States: Onchip Stack: 8
 Offchip Stack: 12

Flags Affected					
Z	N	C	V	VT	ST
0	0	0	0	0	0

75. RET — RETURN FROM SUBROUTINE

Operation: The PC is popped off the top of the stack.

$PC \leftarrow (SP)$
 $SP \leftarrow SP + 2$

Assembly Language Format: RET

Object Code Format: [11110000]

Bytes: 1
 States: Onchip Stack: 12
 Offchip Stack: 16

Flags Affected					
Z	N	C	V	VT	ST
--	--	--	--	--	--

Operation: The PSW is initialized to zero, and the PC is initialized to 2080H. The I/O registers are set to their initial value. Executing this instruction will cause a pulse to appear on the reset pin of the 8096.

PSW \leftarrow 0
PC \leftarrow 2080H

Assembly Language Format: RST

Object Code Format: [11111111]

Bytes: 1
States: 16

Flags Affected					
Z	N	C	V	VT	ST
0	0	0	0	0	0

77. SCALL — SHORT CALL

Operation: The contents of the program counter (the return address) is pushed onto the stack. Then the distance from the end of this instruction to the target label is added to the program counter, effecting the call. The offset from the end of this instruction to the target label must be in the range of -1024 to $+1023$ inclusive.

SP \leftarrow SP - 2
(SP) \leftarrow PC
PC \leftarrow PC + disp (sign-extended to 16 bits)

Assembly Language Format: SCALL cadd

Object Code Format: [00101xxx] [disp-low]

where xxx holds the three high-order bits of displacement.

Bytes: 2
States: Onchip Stack: 13
Offchip Stack: 16

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

78. SETC — SET CARRY FLAG

80. SHLB — SHIFT BYTE LEFT

Operation: The carry flag is set.
 $C \leftarrow 1$

Assembly Language Format: SETC

Object Code Format: [11111001]

Bytes: 1
 States: 4

Flags Affected					
Z	N	C	V	VT	ST
—	—	1	—	—	—

79. SHL — SHIFT WORD LEFT

Operation: The destination (leftmost) word operand is shifted left as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH) inclusive, or as the content of any register. Details on indirect shifts can be found in the Overview. The right bits of the result are filled with zeroes. The last bit shifted out is saved in the carry flag.

Temp \leftarrow (COUNT)
 do while Temp \neq 0
 C \leftarrow High order bit of (DEST)
 (DEST) \leftarrow (DEST) * 2
 Temp \leftarrow Temp - 1
 end_while

Assembly Language Format: SHL wreg, #count
 or
 SHL wreg, breg

Object Code Format: [00001001] [cnt/breg] [wreg]

Bytes: 3
 States: 7 + No. of shifts performed
 note: 0 place shifts take 8 states.

Flags Affected					
Z	N	C	V	VT	ST
✓	?	✓	✓	↑	—

80. SHLB — SHIFT BYTE LEFT

78 SETC — SET CARRY FLAG

Operation: The destination (leftmost) byte operand is shifted left as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH) inclusive, or as the content of any register. Details on indirect shifts can be found in the Overview. The right bits of the result are filled with zeroes. The last bit shifted out is saved in the carry flag.

```
Temp ← (COUNT)
do while Temp <> 0
  C ← High order bit of (DEST)
  (DEST) ← (DEST) * 2
  TEMP ← Temp - 1
end_while
```

Assembly Language Format:

or

SHLB breg, #count

SHLB breg, breg

Object Code Format: [00011001] [cnt/breg] [breg]

Bytes 3

States: 7 + No. of shifts performed

note: 0 place shifts take 8 states.

Flags Affected

Z	N	C	V	VT	ST
✓	?	✓	✓	↑	—

81. SHLL — SHIFT DOUBLE-WORD LEFT

Operation: The destination (leftmost) double-word operand is shifted left as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH) inclusive, or as the content of any register. Details on indirect shifts can be found in the Overview. The right bits of the result are filled with zeroes. The last bit shifted out is saved in the carry flag.

```
Temp ← (COUNT)
do while Temp <> 0
  C ← High order bit of (DEST)
  (DEST) ← (DEST) * 2
  Temp ← Temp - 1
end_while
```

Assembly Language Format: SHLL lreg, #count
or
SHLL lreg, breg

Object Code Format: [00001101] [cnt/breg] [lreg]

Bytes: 3

States: 7 + No. of shifts performed

note: 0 place shifts take 8 states

Flags Affected					
Z	N	C	V	VT	ST
✓	?	✓	✓	↑	—

Operation: The destination (leftmost) word operand is shifted right as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH) inclusive, or as the content of any register. Details on indirect shifts can be found in the Overview. The left bits of the result are filled with zeroes. The last bit shifted out is saved to the carry. The sticky bit flag is cleared at the beginning of the instruction, and set if at any time during the shift a 1 is shifted first into the carry flag, and a further shift cycle occurs.

```
Temp ← (COUNT)
do while Temp <> 0
  C ← Low order bit of (DEST)
  (DEST) ← (DEST) / 2 where / is unsigned division
  Temp ← Temp - 1
end_while
```

Assembly Language Format:

SHR wreg, #count

or

SHR wreg, breg

Object Code Format: [00001000] [cnt/breg] [wreg]

Bytes: 3

States: 7 + No. of shifts performed
note: 0 place shifts take 8 states.

Flags Affected					
Z	N	C	V	VT	ST
✓	0	✓	0	—	✓

Operation: The destination (leftmost) word operand is shifted right as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH) inclusive, or as the content of any register. Details on indirect shifts can be found in the Overview. If the original high order bit value was 0, zeroes are shifted in. If the value was 1, ones are shifted in. The last bit shifted out is saved in the carry. The sticky bit flag is cleared at the beginning of the instruction, and set if at any time during the shift a 1 is shifted first into the carry flag, and a further shift cycle occurs.

```
Temp ← (COUNT)
do while Temp <> 0
  C ← Low order bit of (DEST)
  (DEST) ← (DEST) / 2 where / is signed division
  Temp ← Temp - 1
end_while
```

Assembly Language Format:

SHRA wreg,#count
or
SHRA wreg,breg

Object Code Format: [00001010] [cnt/breg] [wreg]

Bytes: 3
States: 7 + No. of shifts performed
note: 0 place shifts take 8 states.

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	0	—	✓

84. SHRAB — ARITHMETIC RIGHT SHIFT BYTE

Operation: The destination (leftmost) byte operand is shifted right as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH) inclusive, or as the content of any register. Details on indirect shifts can be found in the Overview. If the original high order bit value was 0, zeroes are shifted in. If that value was 1, ones are shifted in. The last bit shifted out is saved in the carry. The sticky bit flag is cleared at the beginning of the instruction, and set if at any time during the shift a 1 is shifted first into the carry flag, and a further shift cycle occurs.

```
Temp ← (COUNT)
do while Temp <> 0
  C, = Low order bit of (DEST)
  (DEST) ← (DEST) / 2 where / is signed division
  Temp ← Temp - 1
end_while
```

Assembly Language Format: SHRAB breg,#count

or

SHRAB breg,breg

Object Code Format: [00011010] [cnt/breg] [breg]

Bytes: 3

States: 7 + No. of shifts performed

note: 0 place shifts take 8 states.

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	0	—	✓

85. SHRAL — ARITHMETIC RIGHT SHIFT DOUBLE-WORD

Operation: The destination (leftmost) double-word operand is shifted right as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH) inclusive, or as the content of any register. Details on indirect shifts can be found in the Overview. If the original high order bit value was 0, zeroes are shifted in. If the value was 1, ones are shifted in. The sticky bit is cleared at the beginning of the instruction, and set if at any time during the shift a 1 is shifted first into the carry flag, and a further shift cycle occurs.

```
Temp ← (COUNT)
do while Temp <> 0
  C ← Low order bit of (DEST)
  (DEST) ← (DEST) / 2 where / is signed division
  Temp ← Temp - 1
end_while
```

Assembly Language Format:

SHRAL lreg, #count

or

SHRAL lreg, breg

Object Code Format: [00001110] [cnt/breg] [lreg]

Bytes: 3

States: 7 + No. of shifts performed

note: 0 place shifts take 8 states.

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	0	—	✓

86. SHRB — LOGICAL RIGHT SHIFT BYTE

Operation: The destination (leftmost) byte operand is shifted right as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH) inclusive, or as the content of any register. Details on indirect shifts can be found in the Overview. The left bits of the result are filled with zeroes. The last bit shifted out is saved in the carry. The sticky bit flag is cleared at the beginning of the instruction, and set if at any time during the shift a 1 is shifted first into the carry flag, and a further shift cycle occurs.

```
Temp ← (COUNT)
do while Temp <> 0
  C ← Low order bit of (DEST)
  (DEST) ← (DEST) / 2 where / is unsigned division
  Temp ← Temp - 1
end_while
```

Assembly Language Format:

```
SHRB breg, #count
or
SHRB breg, breg
```

Object Code Format: [00011000] [cnt/breg] [breg]

Bytes: 3
States: 7 + No. of shifts performed
note: 0 place shifts take 8 states.

Flags Affected					
Z	N	C	V	VT	ST
✓	0	✓	0	—	✓

87. SHRL — LOGICAL RIGHT SHIFT DOUBLE-WORD

Operation: The destination (leftmost) double-word operand is shifted right as many times as specified by the count (rightmost) operand. The count may be specified either as an immediate value in the range of 0 to 15 (0FH) inclusive, or as the content of any register. Details on indirect shifts can be found in the Overview. The left bits of the result are filled with zeroes. The last bit shifted out is saved in the carry. The sticky bit flag is cleared at the beginning of the instruction, and set if at any time during the shift a 1 is shifted first into the carry flag, and a further shift cycle occurs.

```
Temp ← (COUNT)
do while Temp <> 0
  C ← Low order bit of (DEST)
  (DSET) ← (DEST) / 2 where / is unsigned division
  Temp ← Temp - 1
end_while
```

Assembly Language Format:

or

SHRL	lreg, #count	N	Z
SHRL	lreg, breg	—	—

Object Code Format: [00001100] [cnt/breg] [lreg]

Bytes: 3

States: 7 + No. of shifts performed
note: 0 place shifts take 8 states.

Flags Affected

Z	N	C	V	VT	ST
✓	0	✓	0	—	✓

Object Code Format: [00000000] [breg]

Bytes: 2
States: 4

Z	N	C	V	VT	ST
—	—	—	—	—	—

88. SJMP — SHORT JUMP

Operation: The distance from the end of this instruction to the target label is added to the program counter, effecting the jump. The offset from the end of this instruction to the label must be in the range of -1024 to +1023 inclusive.

$PC \leftarrow PC + \text{disp (sign-extended to 16 bits)}$

Assembly Language Format: SJMP label

Object Code Format: [00100xxx] [disp-low]
where xxx holds the three high order bits of the displacement.

Bytes: 2
States: 8

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

89. SKIP — TWO BYTE NO-OPERATION

Operation: Nothing is done. This is actually a two-byte NOP where the second byte can be any value, and is simply ignored. Control passes to the next sequential instruction.

Assembly Language Format: SKIP breg

Object Code Format: [00000000] [breg]

Bytes: 2
States: 4

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

90. ST — STORE WORD

Operation: The value of the leftmost word operand is stored into the rightmost operand.
(DEST) ← (SRC)

Assembly Language Format: ST SRC DST
wreg, waop

Object Code Format: [110000aa] [waop] [wreg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

91. STB — STORE BYTE

Operation: The value of the leftmost byte operand is stored into the rightmost operand.
(DEST) ← (SRC)

Assembly Language Format: STB SRC DST
breg, baop

Object Code Format: [110001aa] [baop] [breg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

92. SUB (Two Operands) — SUBTRACT WORDS

Operation: The source (rightmost) word operand is subtracted from the destination (leftmost) word operand, and the result is stored in the destination. The carry flag is set as complement of borrow.

$(\text{DEST}) \leftarrow (\text{DEST}) - (\text{SRC})$

Assembly Language Format:

SUB DST SRC
wreg, waop

Object Code Format: [011010aa] [waop] [wreg]

Bytes: 2 + BEA

States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

93. SUB (Three Operands) — SUBTRACT WORDS

Operation: The source (rightmost) word operand is subtracted from the second word operand, and the result is stored in the destination (the leftmost operand). The carry flag is set as complement of borrow.

$(\text{DEST}) \leftarrow (\text{SRC1}) - (\text{SRC2})$

Assembly Language Format:

SUB DST SRC1 SRC2,
wreg, wreg, waop

Object Code Format: [010010aa] [waop] [Sweg] [Dwreg]

Bytes: 3 + BEA

States: 5 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

94. SUBB (Two Operands) — SUBTRACT BYTES

Operation: The source (rightmost) byte is subtracted from the destination (leftmost) byte operand, and the result is stored in the destination. The carry flag is set as complement of borrow.

$$(DEST) \leftarrow (DEST) - (SRC)$$

Assembly Language Format:

SUBB DST SRC
breg, baop

Object Code Format: [011110aa] [baop] [breg]

Bytes: 2 + BEA

States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

95. SUBB (Three Operands) — SUBTRACT BYTES

Operation: The source (rightmost) byte operand is subtracted from the second byte operand, and the result is stored in the destination (the leftmost operand). The carry flag is set as complement of borrow.

$$(DEST) \leftarrow (SRC1) - (SRC2)$$

Assembly Language Format:

SUBB DST SRC1 SRC2
breg, Sbreg baop

Object Code Format: [010110aa] [baop] [Sbreg] [Dbreg]

Bytes: 3 + BEA

States: 5 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	✓	✓	↑	—

96. SUBC — SUBTRACT WORDS WITH BORROW

Operation: The source (rightmost) word operand is subtracted from the destination (leftmost) word operand. If the carry flag was clear, 1 is subtracted from the above result. The result replaces the original destination operand. The carry flag is set as complement of borrow.

$$(DEST) \leftarrow (DEST) - (SRC) - (1-C)$$

Assembly Language Format:

SUBC DST SRC
 wreg, waop

Object Code Format: [101010aa] [waop] [wreg]

Bytes: 2 + BEA

States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
↓	↗	↗	↗	↑	—

97. SUBCB — SUBTRACT BYTES WITH BORROW

Operation: The source (rightmost) byte operand is subtracted from the destination (leftmost) byte operand. If the carry flag was clear, 1 is subtracted from the above result. The result replaces the original destination operand. The carry flag is set as complement of borrow.

$$(DEST) \leftarrow (DEST) - (SRC) - (1-C)$$

Assembly Language Format:

SUBCB DST SRC
 breg, baop

Object Code Format: [101110aa] [baop] [breg]

Bytes: 2 + BEA

States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
↓	↗	↗	↗	↑	—

98. TRAP — SOFTWARE TRAP

Operation: This instruction causes an interrupt-call which is vectored through location 2010H. The operation of this instruction is not effected by the state of the interrupt enable flag in the PSW (I). Interrupt-calls cannot occur immediately following this instruction. This instruction is intended for use by Intel provided development tools. These tools will not support user-application of this instruction.

SP ← SP - 2

(SP) ← PC

PC ← (2010H)

Assembly Language Format: This instruction is not supported by revision 1.0 of the 8096 assembly language.

Object Code Format: [11110111]

Bytes:

States

Onchip Stack:

Offchip Stack:

1

21

24

Flags Affected

Z	N	C	V	VT	ST
—	—	—	—	—	—

99. XOR — LOGICAL EXCLUSIVE-OR WORDS

Operation: The source (rightmost) word operand is XORed with the destination (leftmost) word operand. Each bit is set to 1 if the corresponding bit in either the source operand or the destination operand was 1, but not both. The result replaces the original destination operand.

(DEST) ← (DEST) XOR (SRC)

Assembly Language Format:

XOR DST SRC
wreg, waop

Object Code Format: [100001aa] [waop] [wreg]

Bytes: 2 + BEA

States: 4 + CEA

Flags Affected

Z	N	C	V	VT	ST
✓	✓	0	0	—	—

100. XORB — LOGICAL EXCLUSIVE-OR BYTES

Operation: The source (rightmost) byte operand is XORed with the destination (leftmost) byte operand. Each bit is set to 1 if the corresponding bit in either the source operand or the destination operand was 1, but not both. The result replaces the original destination operand.

(DEST) ← (DEST) XOR (SRC)

Assembly Language Format:

	DST	SRC
XORB	breg,	baop

Object Code Format: [100101aa] [baop] [breg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

Flags Affected					
Z	N	C	V	VT	ST
—	—	—	—	—	—

99. XOR — LOGICAL EXCLUSIVE-OR WORDS

Operation: The source (rightmost) word operand is XORed with the destination (leftmost) word operand. Each bit is set to 1 if the corresponding bit in either the source operand or the destination operand was 1, but not both. The result replaces the original destination operand.

(DEST) ← (DEST) XOR (SRC)

Assembly Language Format:

	DST	SRC
XOR	wreg,	wreg

Object Code Format: [10001aa] [wreg] [wreg]

Bytes: 2 + BEA
States: 4 + CEA

Flags Affected					
Z	N	C	V	VT	ST
✓	✓	0	0	—	—

MCS[®]-96 Hardware Design Information

17

MCS-96 Hardware Design Information

17

MCS®-96 HARDWARE DESIGN INFORMATION

OVERVIEW

This Chapter of the manual is devoted to the hardware engineer. All of the information you need to connect the correct pin to the correct external circuit is provided. Many of the special function pins have different characteristics which are under software control, therefore, it is necessary to define the system completely before the hardware is wired-up.

Frequently within this chapter a specification for a current, voltage, or time period is referred to; the values provided are to be used as an approximation only. The exact specification can be found in the latest data sheet for the particular part and temperature range that is being used.

1.0 REQUIRED HARDWARE CONNECTIONS

Although the 8096BH is a single-chip microcontroller, it still requires several external connections to make it work. Power must be applied, a clock source provided, and some form of reset circuitry must be present. We will look at each of these areas of circuitry separately. Figure 6 shows the connections that are needed for a single-chip system.

1.1 Power Supply Information

Power for the 8096BH flows through six pins; they are: three positive voltage pins— V_{CC} (digital), V_{REF} (Port 0 digital I/O and A/D power), V_{PD} (power down mode); and three common returns—two V_{SS} pins and one $ANGND$ pin. All six of these pins must be connected on the 8096BH for normal operation. The V_{CC} pin, V_{REF} pin and V_{PD} pin should be tied to 5 volts. The two V_{SS} pins and the $ANGND$ pin must be grounded. When the analog to digital converter is being used it may be desirable to connect the V_{REF} pin to a separate power supply, or at least a separate power supply line.

The three common return pins should be connected at the chip with as short a lead as possible to avoid problems due to voltage drops across the wiring. There should be no measurable voltage difference between V_{SS1} and V_{SS2} . The two V_{SS} pins and the $ANGND$ pin must all be nominally at 0 volts. The maximum current drain of the 8096BH is around 180 mA, with all lines unloaded.

When the analog converter is being used, clean, stable power must be provided to the analog section of the chip to assure highest accuracy. To achieve this, it may be desirable to separate the analog power supply from the digital power supply. The V_{REF} pin supplies the digital circuitry in the A/D converter and provides the 5 volt reference to the analog portion of the converter. V_{REF} and $ANGND$ must be connected even if the A/D converter is not used. More information on the analog power supply is in Section 3.1.

1.2 Other Needed Connections

Several other connections are needed to configure the 8096BH. In normal operation the following pins should be connected to the indicated power supply.

Pin		Power Supply
8X9XBH	8X9X	
NMI	NMI	V_{CC}
\overline{EA}	\overline{EA}	V_{CC}
	TEST	V_{CC} (to allow internal execution)
		V_{SS} (to force external execution)

Although the \overline{EA} pin has an internal pulldown, it is best to tie this pin to the desired level. This will prevent induced noise from disturbing the system. With the exception of 8X9X devices, raising \overline{EA} to +12.75 volts will place an 8096BH in a special operating mode designed for programming and program memory verification (see Section 10).

1.3 Oscillator Information

The 8096BH requires a clock source to operate. This clock can be provided to the chip through the XTAL1 input or the on-chip oscillator can be used. The frequency of operation is from 6 MHz to 12 MHz.

The on-chip circuitry for the 8096BH oscillator is a single stage linear inverter as shown in Figure 1. It is intended for use as a crystal-controlled, positive reactance oscillator with external connections as shown in Figure 2. In this application, the crystal is being operated in its fundamental response mode as an inductive

reactance in parallel resonance with shunt capacitance external to the crystal.

The crystal specifications and capacitance values (C1 and C2 in Figure 2) are not critical. Thirty pF can be used in these positions at any frequency with good quality crystals. For 0.5% frequency accuracy, the crystal frequency can be specified at series resonance or

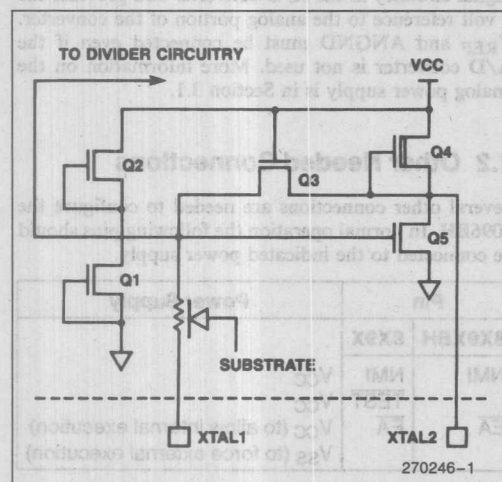


Figure 1. 8096BH Oscillator Circuit

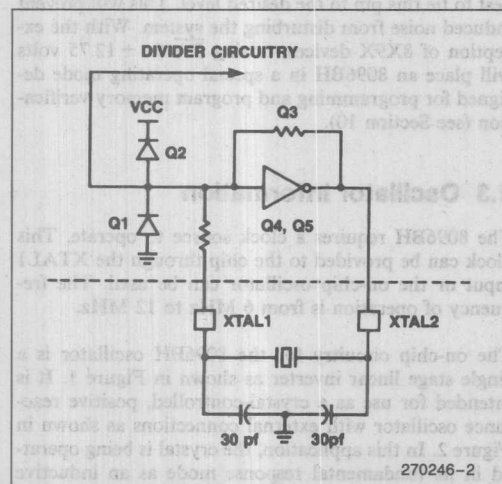


Figure 2. Crystal Oscillator Circuit

for parallel resonance with any load capacitance. (In other words, for that degree of frequency accuracy, the load capacitance simply doesn't matter.) For 0.05% frequency accuracy the crystal frequency should be specified for parallel resonance with 25 pF load capacitance, if C1 and C2 are 30 pF.

A more in-depth discussion of crystal specifications and the selection of values for C1 and C2 can be found in the Intel Application Note, AP-155, "Oscillators for Microcontrollers."

To drive the 8096BH with an external clock source, apply the external clock signal to XTAL1 and let XTAL2 float. An example of this circuit is shown in Figure 3. The required voltage levels on XTAL1 are specified in the data sheet. The signal on XTAL1 must be clean with good solid levels.

It is important that the minimum high and low times are met to avoid having the XTAL1 pin in the transition range for long periods of time. The longer the signal is in the transition region, the higher the probability that an external noise glitch could be seen by the clock generator circuitry. Noise glitches on the 8096BH internal clock lines will cause unreliable operation.

The clock generator provides a 3 phase clock output from the XTAL1 pin input. Figure 4 shows the waveforms of the major internal timing signals.

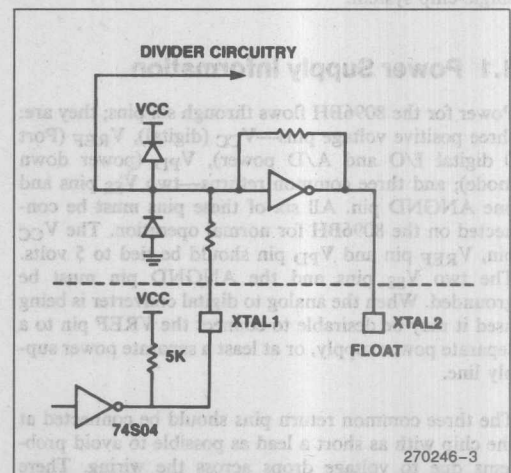


Figure 3. External Clock Drive

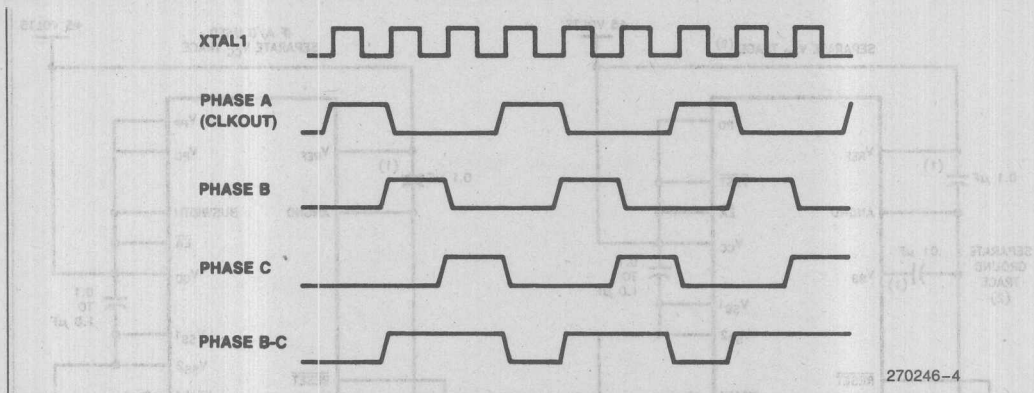


Figure 4. Internal Timings

1.4 Reset Information

In order for the 8096BH to function properly it must be reset. This is done by holding the $\overline{\text{RESET}}$ pin low for at least 2 state times after the power supply is within tolerance and the oscillator has stabilized.

On 8X9X devices the $\overline{\text{RESET}}$ pin must be held low long enough for the power supply, oscillator and back-bias generator to stabilize. Typically, the back-bias generator requires one millisecond to stabilize.

After the $\overline{\text{RESET}}$ pin is brought high, a ten state reset sequence is executed. During this time, the Chip Configuration Byte (CCB) is read from location 2018H and written to the 8096BH Chip Configuration Register (CCR). If the voltage on the $\overline{\text{EA}}$ pin selects the internal/external execution mode the CCB is read from in-

ternal ROM/EPROM. If the voltage on the $\overline{\text{EA}}$ pin selects the external execution only mode the CCB is read from external memory. See Figure 5.

On 8X9X devices, the CCB read does not occur, and ALE is high while $\overline{\text{RESET}}$ is held low.

There are several ways to provide a good reset to an 8096BH, the simplest being just to connect a capacitor from the reset pin to ground. The capacitor should be on the order of 2 microfarads for every millisecond of reset time required. This method will only work if the rise time of V_{CC} is fast and the total reset time is less than around 50 milliseconds. It also may not work if the $\overline{\text{RESET}}$ pin is to be used to reset other parts on the board. An 8096BH with the minimum required connections is shown in Figure 6.

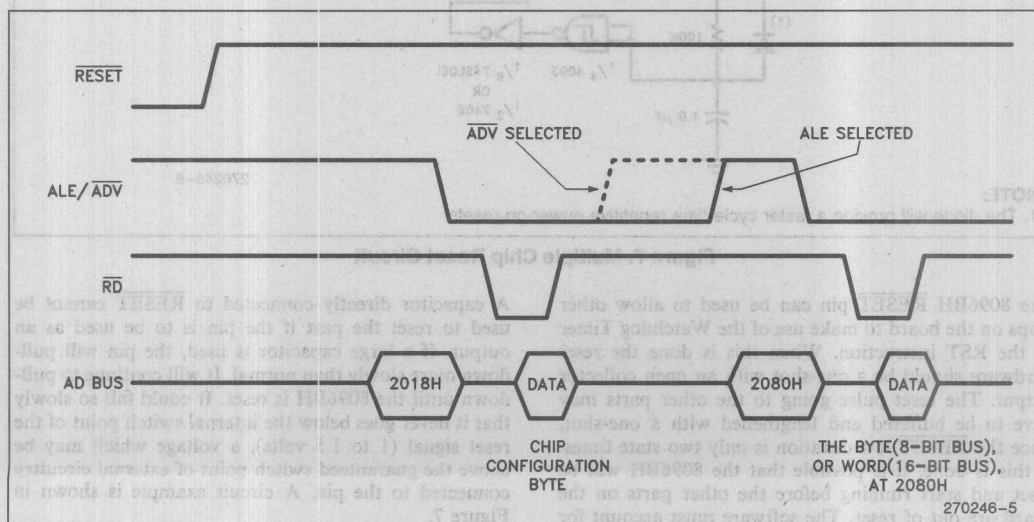


Figure 5. Reset Sequence

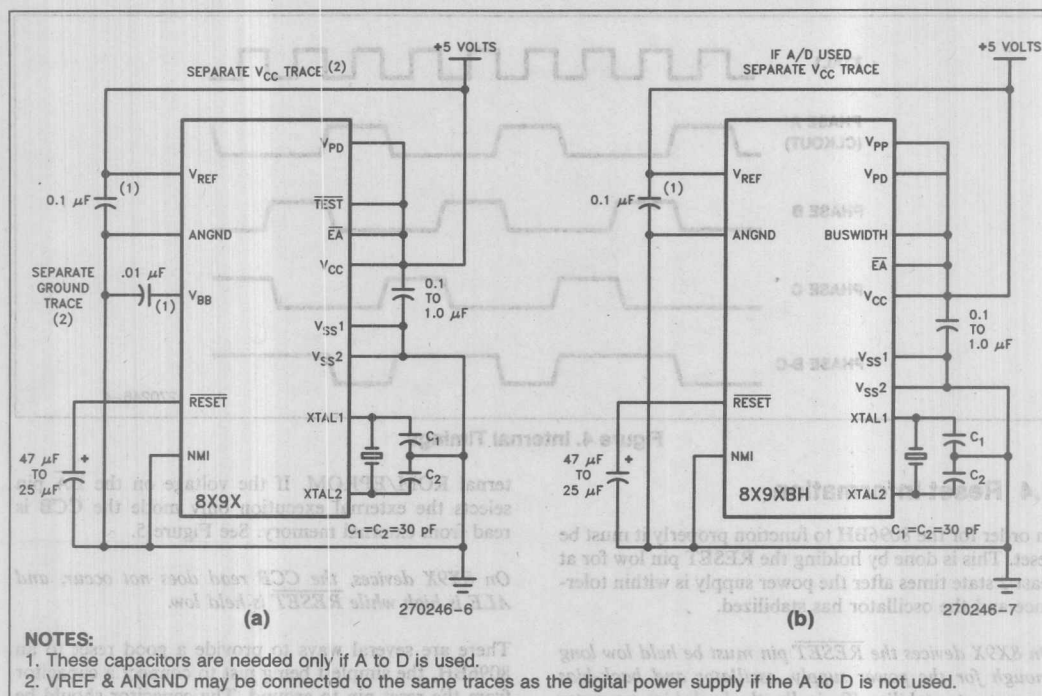


Figure 6. Minimum Hardware Connections

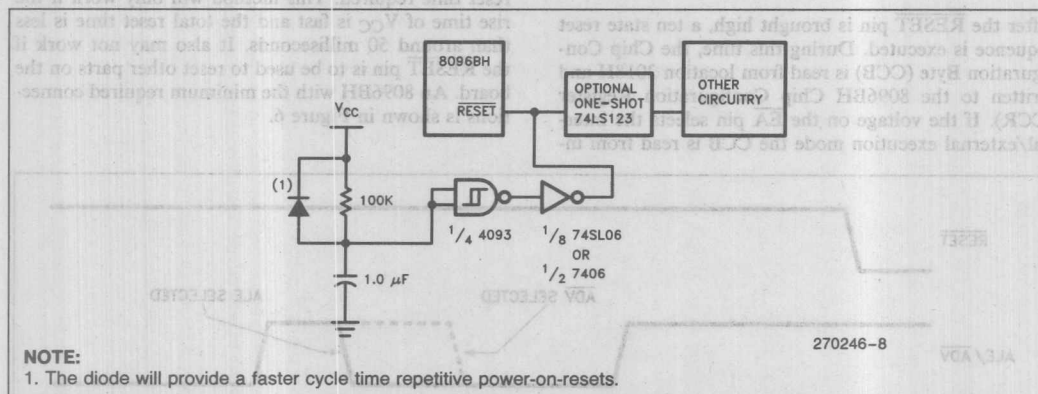


Figure 7. Multiple Chip Reset Circuit

The 8096BH RESET pin can be used to allow other chips on the board to make use of the Watchdog Timer or the RST instruction. When this is done the reset hardware should be a one-shot with an open collector output. The reset pulse going to the other parts may have to be buffered and lengthened with a one-shot, since the RESET low duration is only two state times. If this is done, it is possible that the 8096BH will be reset and start running before the other parts on the board are out of reset. The software must account for this possible problem.

A capacitor directly connected to RESET cannot be used to reset the part if the pin is to be used as an output. If a large capacitor is used, the pin will pull-down more slowly than normal. It will continue to pull-down until the 8096BH is reset. It could fall so slowly that it never goes below the internal switch point of the reset signal (1 to 1.5 volts), a voltage which may be above the guaranteed switch point of external circuitry connected to the pin. A circuit example is shown in Figure 7.

1.5 Sync Mode

If **RESET** is brought high at the same time as or just after the rising edge of XTAL1, the part will start executing the 10 state time RST instruction exactly $6\frac{1}{2}$ XTAL1 cycles later. This feature can be used to synchronize several MCS-96 devices. A diagram of a typical connection is shown in Figure 8. It should be noted that parts that start in sync may not stay that way, due to propagation delays which may cause the synchronized parts to receive signals at slightly different times.

1.6 Disabling the Watchdog Timer

The Watchdog Timer will pull the **RESET** pin low when it overflows. See Figure 9. If the pin is being externally held above the low going threshold, the pull-down transistor will remain on indefinitely. This means that once the watchdog overflows, the part must be reset or **RESET** must be held high indefinitely. Just

resetting the Watchdog Timer in software will not clear the flip-flop which keeps the **RESET** pulldown on.

The pulldown is capable of sinking on the order of 30 milliamps if it is held at 2.0 volts. This amount of current may cause some long term reliability problems due to localized chip heating. For this reason, parts that will be used in production should never have had the Watchdog Timer over-ridden for more than a second or two.

Whenever the reset pin is being pulled high while the pulldown is on, it should be through a resistor that will limit the voltage on **RESET** to 2.5 volts and the current through the pin to 40 milliamps.

If it is necessary to disable the Watchdog Timer for more than a brief test the software solution of never initiating the timer should be used. See Section 14 in the Architecture Chapter.

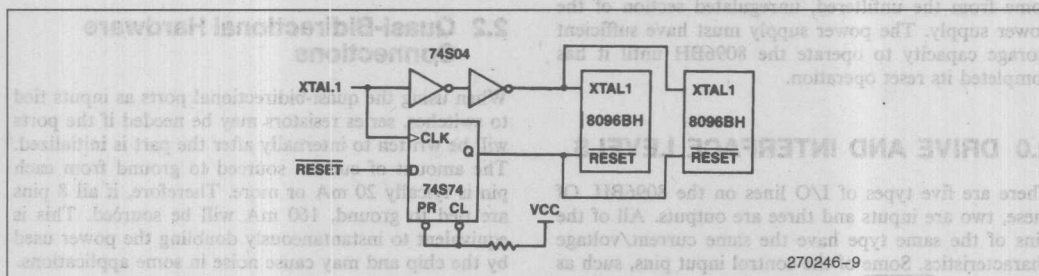


Figure 8. Reset Sync Mode

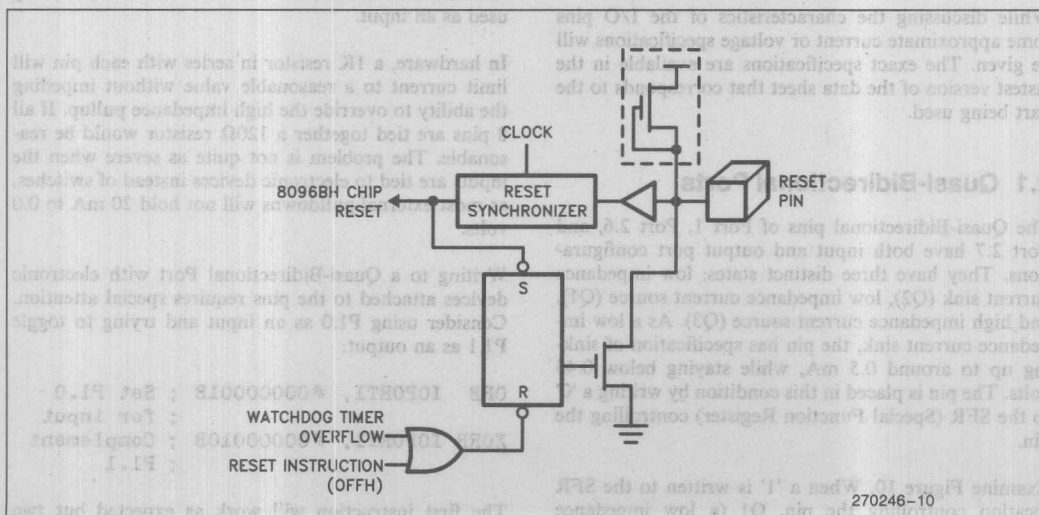


Figure 9. Reset Logic

1.7 Power Down Circuitry

Battery backup can be provided on the 8096BH with a 1 mA current drain at 5 volts. This mode will hold locations OF0H through OFFH valid as long as the power to the V_{DD} pin remains on. The required timings to put the part into power-down and an overview of this mode are given in Section 4.2 in the MCS-96 Architecture Chapter.

A 'key' can be written into power-down RAM while the part is running. This key can be checked on reset to determine if it is a start-up from power-down or a complete cold start. In this way the validity of the power-down RAM can be verified. The length of this key determines the probability that this procedure will work, however, there is always a statistical chance that the RAM will power up with a replica of the key.

Under most circumstances, the power-fail indicator which is used to initiate a power-down condition must come from the unfiltered, unregulated section of the power supply. The power supply must have sufficient storage capacity to operate the 8096BH until it has completed its reset operation.

2.0 DRIVE AND INTERFACE LEVELS

There are five types of I/O lines on the 8096BH. Of these, two are inputs and three are outputs. All of the pins of the same type have the same current/voltage characteristics. Some of the control input pins, such as XTAL1 and RESET, may have slightly different characteristics. These pins are discussed in Section 1.

While discussing the characteristics of the I/O pins some approximate current or voltage specifications will be given. The exact specifications are available in the latest version of the data sheet that corresponds to the part being used.

2.1 Quasi-Bidirectional Ports

The Quasi-Bidirectional pins of Port 1, Port 2.6, and Port 2.7 have both input and output port configurations. They have three distinct states; low impedance current sink (Q2), low impedance current source (Q1), and high impedance current source (Q3). As a low impedance current sink, the pin has specification of sinking up to around 0.5 mA, while staying below 0.45 volts. The pin is placed in this condition by writing a '0' to the SFR (Special Function Register) controlling the pin.

Examine Figure 10. When a '1' is written to the SFR location controlling the pin, Q1 (a low impedance MOSFET pullup) is turned on for one state time, then it is turned off and the depletion pullup holds the line at

a logical '1' state. The low-impedance pullup is used to shorten the rise time of the pin, and has current source capability on the order of 100 times that of the depletion pullup.

While the depletion mode pullup is the only device on, the pin may be used as an input with a leakage of around 100 microamps from 0.45 volts to VCC. It is ideal for use with TTL or CMOS chips and may even be used directly with switches. However if the switch option is used, certain precautions should be taken. It is important to note that any time the pin is read, the value returned will be the value on the pin, not the value placed in the control register. This could cause logical operations made directly on these pins to inadvertently write a 0 to pins being used as inputs. In order to perform logical operations on a port where a quasi-bidirectional pin is an input, it is necessary to guarantee that the bit associated with the input pin is always a one when writing to the port.

2.2 Quasi-Bidirectional Hardware Connections

When using the quasi-bidirectional ports as inputs tied to switches, series resistors may be needed if the ports will be written to internally after the part is initialized. The amount of current sourced to ground from each pin is typically 20 mA or more. Therefore, if all 8 pins are tied to ground, 160 mA will be sourced. This is equivalent to instantaneously doubling the power used by the chip and may cause noise in some applications.

This potential problem can be solved in hardware or software. In software, never write a zero to a pin being used as an input.

In hardware, a 1K resistor in series with each pin will limit current to a reasonable value without impeding the ability to override the high impedance pullup. If all 8 pins are tied together a 120Ω resistor would be reasonable. The problem is not quite as severe when the inputs are tied to electronic devices instead of switches, as most external pulldowns will not hold 20 mA to 0.0 volts.

Writing to a Quasi-Bidirectional Port with electronic devices attached to the pins requires special attention. Consider using P1.0 as an input and trying to toggle P1.1 as an output:

```
ORB  IOPORT1, #00000001B ; Set P1.0
                                ; for input
XORB IOPORT1, #00000010B ; Complement
                                ; P1.1
```

The first instruction will work as expected but two problems can occur when the second instruction executes. The first is that even though P1.1 is being driven

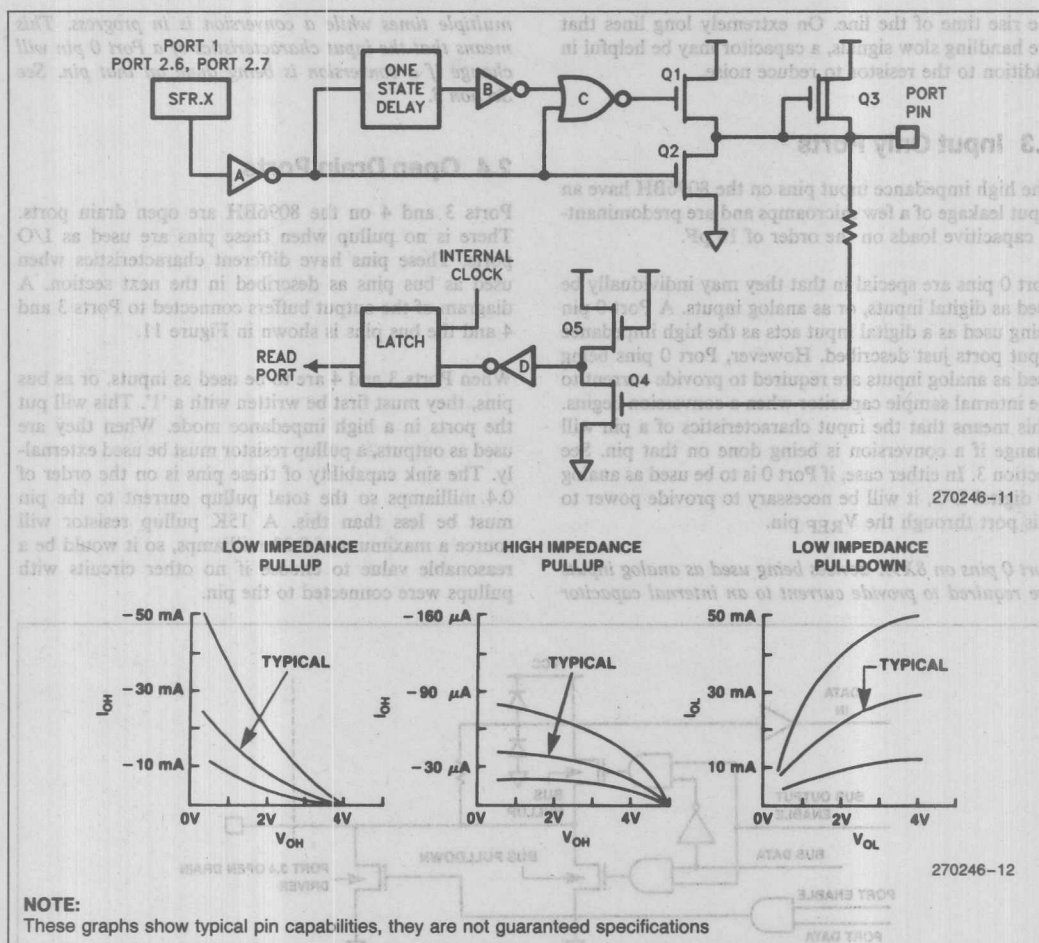


Figure 10. Quasi-Bidirectional Port

high by the 8096 it is possible that it is being held low externally. This typically happens when the port pin is used to drive the base of an NPN transistor which in turn drives whatever there is in the outside world which needs to be toggled. The base of the transistor will clamp the port pin to the transistor's V_{be} above ground, typically 0.7V. The 8096 will input this value as a zero even if a one has been written to the port pin. When this happens the XORB instruction will always write a one to the port pin's SFR and the pin will not toggle.

The second problem, which is related to the first, is that if P1.0 happens to be driven to a zero when Port 1 is read by the XORB instruction, then the XORB will write a zero to P1.0 and it will no longer be useable as an input.

The first situation can best be solved by the external

driver design. A series resistor between the port pin and the base of the transistor often works by bringing up the voltage present on the port pin. The second case can be taken care of in the software fairly easily:

```
LDB AL, IOPORT1
XORB AL, #010B
ORB AL, #001B
STB AL, IOPORT1
```

A software solution to both cases is to keep a byte in RAM as an image of the data to be output to the port; any time the software wants to modify the data on the port it can then modify the image byte and copy it to the port.

If a switch is used on a long line connected to a quasi-bidirectional pin, a pullup resistor is recommended to reduce the possibility of noise glitches and to decrease

the rise time of the line. On extremely long lines that are handling slow signals, a capacitor may be helpful in addition to the resistor to reduce noise.

2.3 Input Only Ports

The high impedance input pins on the 8096BH have an input leakage of a few microamps and are predominantly capacitive loads on the order of 10 pF.

Port 0 pins are special in that they may individually be used as digital inputs, or as analog inputs. A Port 0 pin being used as a digital input acts as the high impedance input ports just described. However, Port 0 pins being used as analog inputs are required to provide current to the internal sample capacitor when a conversion begins. This means that the input characteristics of a pin will change if a conversion is being done on that pin. See Section 3. In either case, if Port 0 is to be used as analog or digital I/O, it will be necessary to provide power to this port through the V_{REF} pin.

Port 0 pins on 8X9X devices being used as analog inputs are required to provide current to an internal capacitor

multiple times while a conversion is in progress. This means that the input characteristics of a Port 0 pin will change if a conversion is being done on that pin. See Section 3.

2.4 Open Drain Ports

Ports 3 and 4 on the 8096BH are open drain ports. There is no pullup when these pins are used as I/O ports. These pins have different characteristics when used as bus pins as described in the next section. A diagram of the output buffers connected to Ports 3 and 4 and the bus pins is shown in Figure 11.

When Ports 3 and 4 are to be used as inputs, or as bus pins, they must first be written with a '1'. This will put the ports in a high impedance mode. When they are used as outputs, a pullup resistor must be used externally. The sink capability of these pins is on the order of 0.4 milliamps so the total pullup current to the pin must be less than this. A 15K pullup resistor will source a maximum of 0.33 milliamps, so it would be a reasonable value to choose if no other circuits with pullups were connected to the pin.

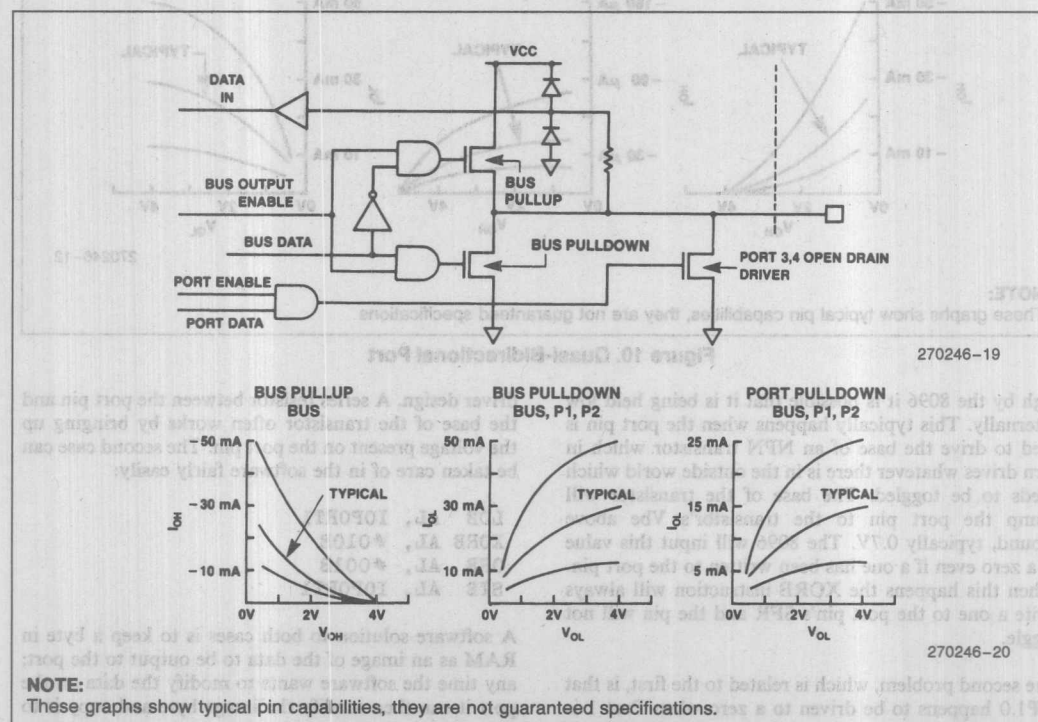


Figure 11. Bus and Port 3 and 4 Pins

2.5 HSO Pins, Control Outputs and Bus Pins

The control outputs and HSO pins have output buffers with the same output characteristics as those of the bus pins. Included in the category of control outputs are: TXD, RXD (in Mode 0), PWM, CLKOUT, ALE, BHE, RD, and WR. The bus pins have 3 states: output high, output low, and high impedance input. As a high output, the pins are specified to source around 200 μ A to 2.4 volts, but the pins can source on the order of ten times that value in order to provide the fast rise times. When used as a low output, the pins can sink around 2 mA at 0.45 volts, and considerably more as the voltage increases. When in the high impedance state, the pin acts as a capacitive load with a few microamps of leakage. Figure 11 shows the internal configuration of a bus pin.

3.0 ANALOG INPUTS

The on-chip A/D converter of the 8096BH can be used to digitize analog inputs while analog outputs can be

generated with either the chip's PWM output or HSO unit. This section describes the analog input suggestions. See Section 4 for analog output.

The 8096BH's Integrated A/D converter includes an eight channel analog multiplexer, sample-and-hold circuit and 10-bit analog to digital converter (Figure 12). The 8096BH can therefore select one of eight analog inputs to convert, sample-and-hold the input voltage and convert the voltage into a digital value. Each conversion takes 22 microseconds, including the time required for the sample-and-hold (with XTAL1 = 12 MHz). The method of conversion is successive approximation.

Section 3.6 contains the definitions of numerous terms used in connection with the A/D converter.

The A/D converter of 8X9X devices does not contain a Sample-and-Hold and has a conversion time of 42 μ s (12 MHz on XTAL1). Section 3.5 discusses the differences.

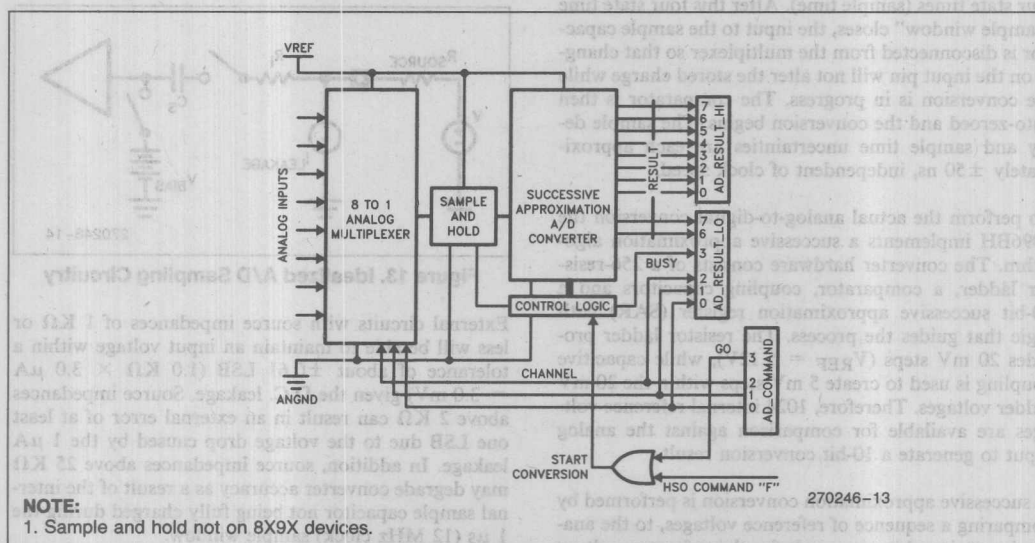


Figure 12. A/D Converter Block Diagram

3.1 A/D Overview

The conversion process is initiated by the execution of HSO command 0FH, or by writing a one to the GO Bit in the A/D Control Register. Either activity causes a start conversion signal to be sent to the A/D converter control logic. If an HSO command was used, the conversion process will begin when Timer 1 increments. This aids applications attempting to approach spectrally pure sampling, since successive samples spaced by equal Timer 1 delays will occur with a variance of about ± 50 ns (assuming a stable clock on XTAL1). However, conversions initiated by writing a one to the ADCON register GO Bit will start within three state times after the instruction has completed execution resulting in a variance of about $0.75 \mu\text{s}$ (XTAL1 = 12 MHz).

Once the A/D unit receives a start conversion signal, there is a one state time delay before sampling (sample delay) while the successive approximation register is reset and the proper multiplexer channel is selected. After the sample delay, the multiplexer output is connected to the sample capacitor and remains connected for four state times (sample time). After this four state time "sample window" closes, the input to the sample capacitor is disconnected from the multiplexer so that changes on the input pin will not alter the stored charge while the conversion is in progress. The comparator is then auto-zeroed and the conversion begins. The sample delay and sample time uncertainties are each approximately ± 50 ns, independent of clock speed.

To perform the actual analog-to-digital conversion the 8096BH implements a successive approximation algorithm. The converter hardware consists of a 256-resistor ladder, a comparator, coupling capacitors and a 10-bit successive approximation register (SAR) with logic that guides the process. The resistor ladder provides 20 mV steps ($V_{\text{REF}} = 5.12\text{V}$), while capacitive coupling is used to create 5 mV steps within the 20 mV ladder voltages. Therefore, 1024 internal reference voltages are available for comparison against the analog input to generate a 10-bit conversion result.

A successive approximation conversion is performed by comparing a sequence of reference voltages, to the analog input, in a binary search for the reference voltage that most closely matches the input. The $\frac{1}{2}$ full scale reference voltage is the first tested. This corresponds to a 10-bit result where the most significant bit is zero, and all other bits are ones (0111.1111.11b). If the analog input was less than the test voltage, bit 10 of the SAR is left a zero, and a new test voltage of $\frac{1}{4}$ full scale (0011.1111.11b) is tried. If this test voltage was lower than the analog input, bit 9 of the SAR is set and bit 8 is cleared for the next test (0101.1111.11b). This binary search continues until 10 tests have occurred, at which time the valid 10-bit conversion result resides in the SAR where it can be read by software.

The total number of state times required is 88 for a 10-bit conversion. Attempting to short-cycle the 10-bit conversion process by reading A/D results before the done bit is set is not recommended.

3.2 A/D Interface Suggestions

The external interface circuitry to an analog input is highly dependent upon the application, and can impact converter characteristics. In the external circuit's design, important factors such as input pin leakage, sample capacitor size and multiplexer series resistance from the input pin to the sample capacitor must be considered.

For the 8096BH, these factors are idealized in Figure 13. The external input circuit must be able to charge a sample capacitor (C_S) through a series resistance (R_I) to an accurate voltage given a D.C. leakage (I_L). On the 8096BH, C_S is around 2 pF, R_I is around 5 K Ω and I_L is specified as 3 μA maximum. In determining the necessary source impedance R_S , the value of V_{BIAS} is not important.

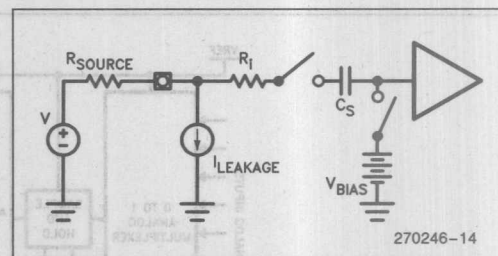


Figure 13. Idealized A/D Sampling Circuitry

External circuits with source impedances of 1 K Ω or less will be able to maintain an input voltage within a tolerance of about ± 0.61 LSB (1.0 K $\Omega \times 3.0 \mu\text{A} = 3.0$ mV) given the D.C. leakage. Source impedances above 2 K Ω can result in an external error of at least one LSB due to the voltage drop caused by the 1 μA leakage. In addition, source impedances above 25 K Ω may degrade converter accuracy as a result of the internal sample capacitor not being fully charged during the 1 μs (12 MHz clock) sample window.

It is important to note that source impedance requirements relax if an external capacitor of sufficient size is attached directly to the analog input pin. Since the internal sample capacitor is around 2.0 pF, an external 0.005 μF capacitor (2048×2.0 pF) should provide an accurate input voltage to ± 0.5 LSB. If there is leakage on the capacitor, the value of the capacitor must be increased to compensate for the leakage. For example, assuming just the 3 μA D.C. leakage caused by the 8096BH, 0.6 mV (less than 0.15 LSB) will be lost from a 0.005 μF capacitor in 1 μs . Therefore, the capacitor

connected externally to the pin should be at least 0.005 μ F if the source impedance is too large to provide the needed accuracy on its own. However, if the external signal changes slowly, it is recommended that the largest acceptable capacitance be used, given the input signal frequency.

Placing an external capacitor on each analog input will also reduce the sensitivity to noise, as the capacitor combines with series resistance in the external circuit to form a low-pass filter. In practice, one should include a small series resistance prior to the external capacitor on the analog input pin and choose the largest capacitor value practical, given the frequency of the signal being converted. This provides a low-pass filter on the input, while the resistor will also limit input current during over-voltage conditions.

Figure 14 shows a simple analog interface circuit based upon the discussion above. The circuit in the figure also provides limited protection against over-voltage conditions on the analog input. Should the input voltage inappropriately drop significantly below ground, diode D2 will forward bias at about 0.8 DCV. Since the specification of the pin has an absolute maximum low voltage rating of -0.3 DCV, this will leave about 0.5 DCV across the 270 Ω resistor, or about 2.0 mA of current. This should limit current to a safe amount. *However, before any circuit is used in an actual application, it should be thoroughly analyzed for applicability to the specific problem at hand.*

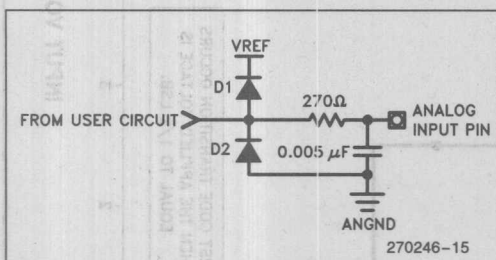


Figure 14. Suggested A/D Input Circuit

3.3 Analog References

Reference supply levels strongly influence the absolute accuracy of the conversion. For this reason, it is recommended that the ANGND pin be tied to the two V_{SS} pins as close to the chip as possible with minimum trace length. Bypass capacitors should also be used between V_{REF} and ANGND. ANGND should be within about a tenth of a volt V_{SS} . V_{REF} should be well regulated and used only for the A/D converter. The V_{REF} supply can be between 4.5V and 5.5V and needs to be able to source around 5 mA. Figure 6 shows all of these connections.

Note that if only ratiometric information is desired, V_{REF} can be connected to V_{CC} . In addition, V_{REF}

and ANGND must be connected even if the A/D converter is not being used. Remember that Port 0 receives its power from the V_{REF} and ANGND pins even when it is used as digital I/O.

3.4 The A/D Transfer Function

The conversion result is a 10-bit ratiometric representation of the input voltage, so the numerical value obtained from the conversion will be:

$$\text{INT} [1023 \times (V_{IN} - \text{ANGND}) / (V_{REF} - \text{ANGND})].$$

This produces a stair-stepped transfer function when the output code is plotted versus input voltage (see Figure 15). The resulting digital codes can be taken as simple ratiometric information, or they can be used to provide information about absolute voltages or relative voltage changes on the inputs. The more demanding the application is on the A/D converter, the more important it is to fully understand the converter's operation. For simple applications, knowing the absolute error of the converter is sufficient. However, closing a servo-loop with analog inputs necessitates a detailed understanding of an A/D converter's operation and errors.

The errors inherent in an analog-to-digital conversion process are many: quantizing error; zero offset; full-scale error; differential non-linearity; and non-linearity. These are "transfer function" errors related to the A/D converter. In addition, converter temperature drift, V_{CC} rejection, sample-hold feedthrough, multiplexer off-isolation, channel-to-channel matching and random noise should be considered. Fortunately, one "Absolute Error" specification is available which describes the sum total of all deviations between the actual conversion process and an ideal converter. However, the various sub-components of error are important in many applications. These error components are described in Section 3.5 and in the text below where ideal and actual converters are compared.

An unavoidable error simply results from the conversion of a continuous voltage to an integer digital representation. This error is called quantizing error, and is always ± 0.5 LSB. Quantizing error is the only error seen in a perfect A/D converter, and is obviously present in actual converters. Figure 15 shows the transfer function for an ideal 3-bit A/D converter (i.e. the Ideal Characteristic).

Note that in Figure 15 the Ideal Characteristic possesses unique qualities: its first code transition occurs when the input voltage is 0.5 LSB; its full-scale code transition occurs when the input voltage equals the full-scale reference minus 1.5 LSB; and its code widths are all exactly one LSB. These qualities result in a digitization without offset, full-scale or linearity errors. In other words, a perfect conversion.

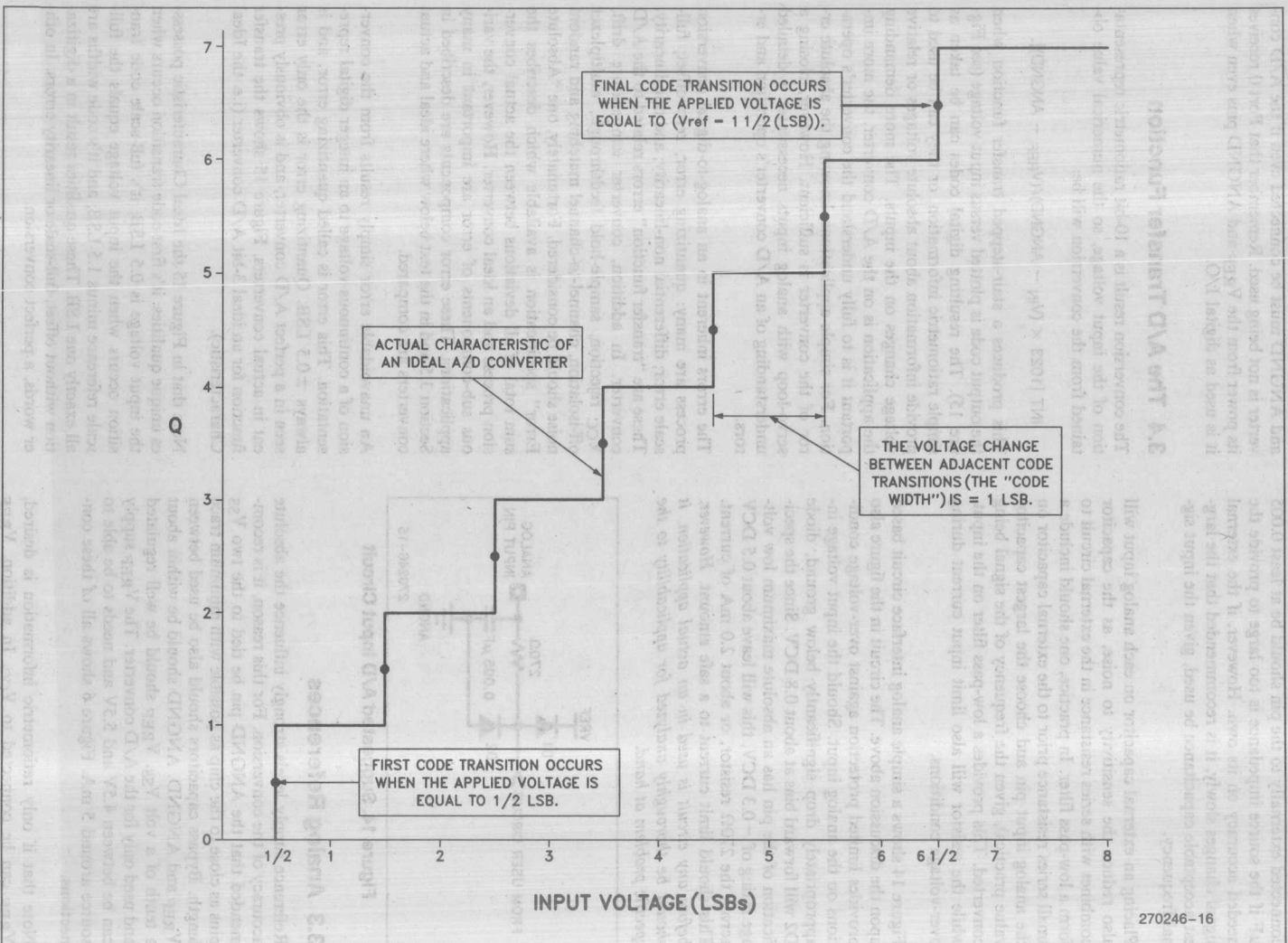


Figure 15. Ideal A/D Characteristic

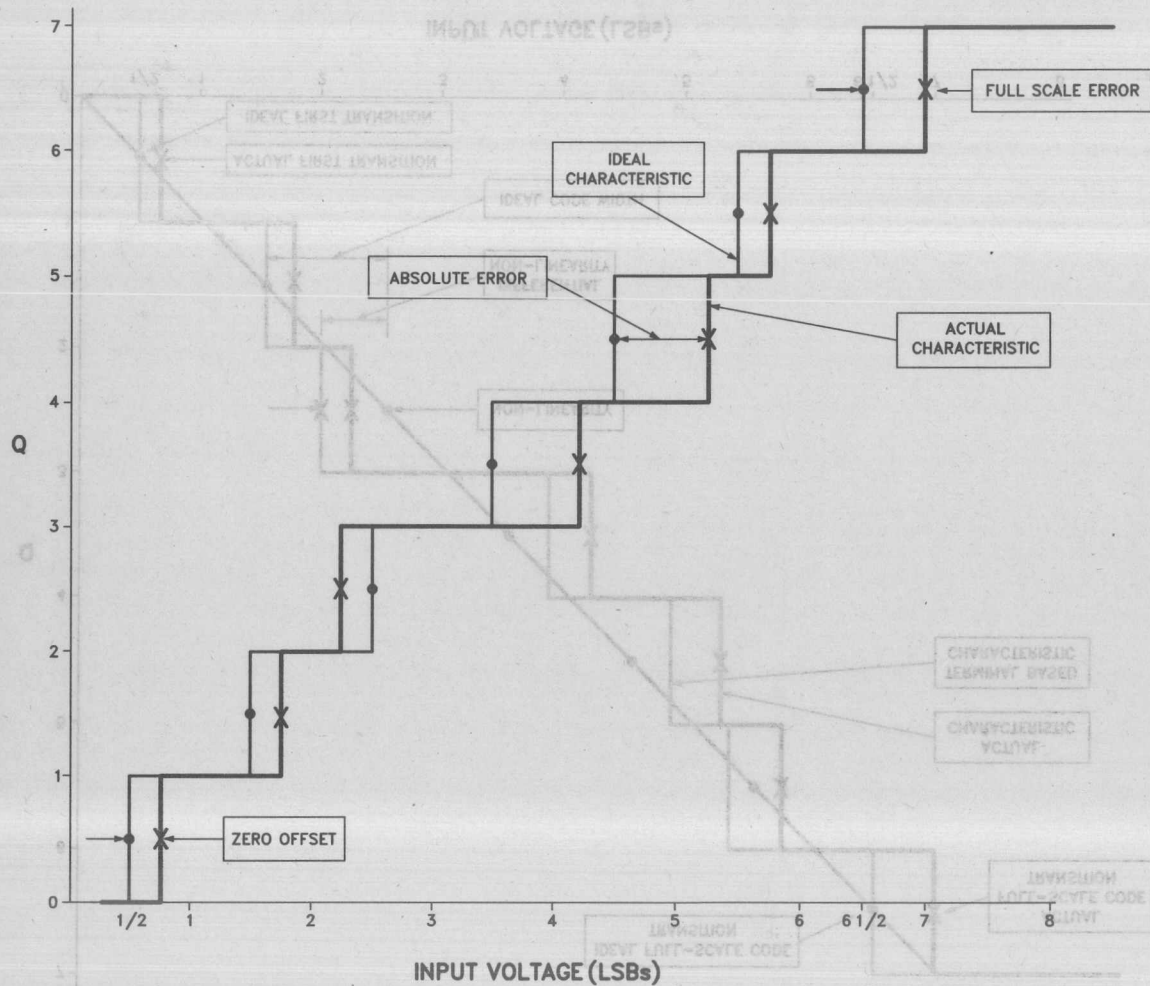
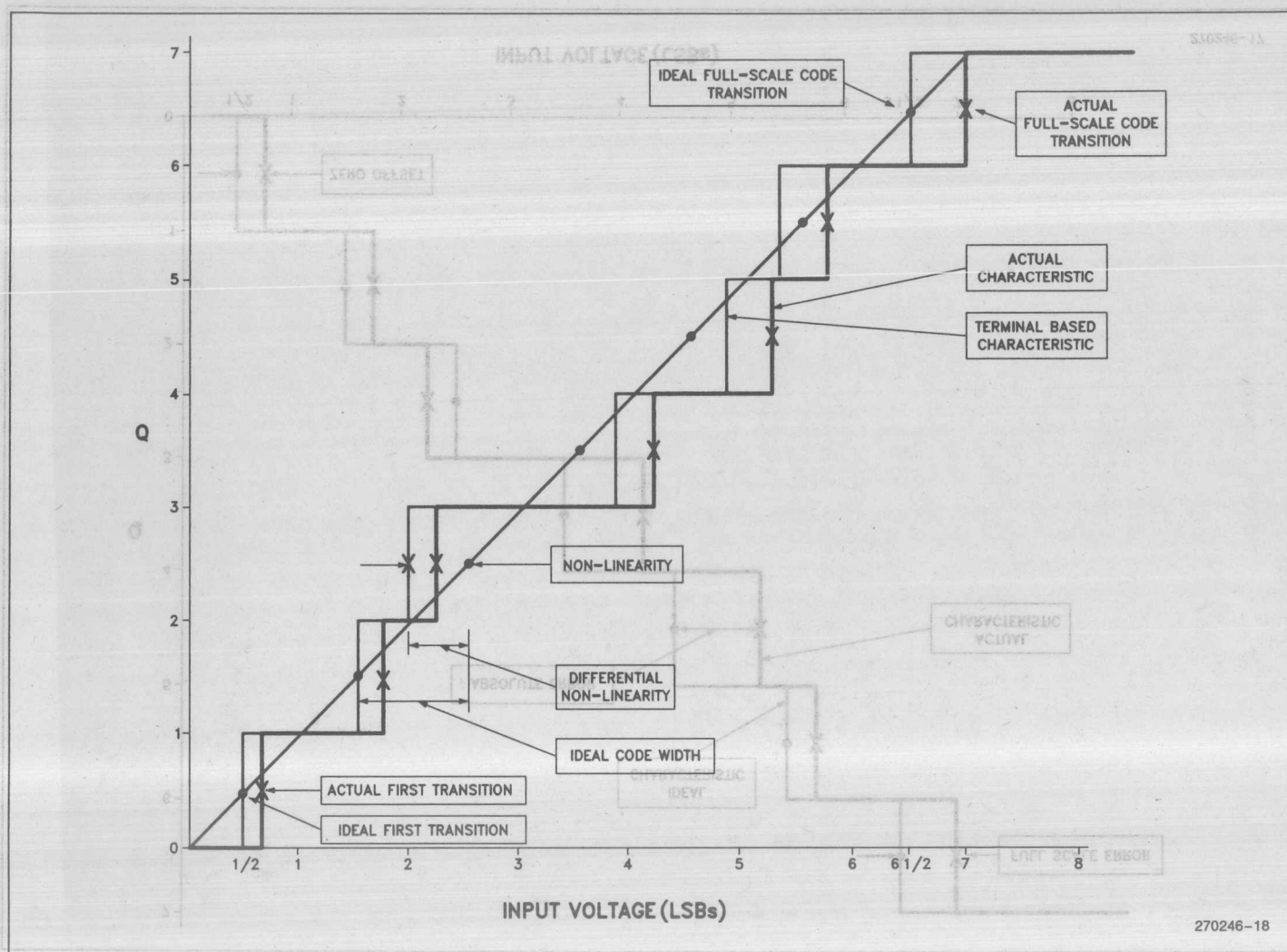


Figure 16. Actual and Ideal Characteristics



270246-18

Figure 16 shows an Actual Characteristic of a hypothetical 3-bit converter, which is not perfect. When the Ideal Characteristic is overlaid with the imperfect characteristic, the actual converter is seen to exhibit errors in the location of the first and final code transitions and code widths. The deviation of the first code transition from ideal is called "zero offset", and the deviation of the final code transition from ideal is "full-scale error". The deviation of the code widths from ideal causes two types of errors. Differential Non-Linearity and Non-Linearity. Differential Non-Linearity is a local linearity error measurement, whereas Non-Linearity is an overall linearity error measure.

Differential Non-Linearity is the degree to which actual code widths differ from the ideal one LSB width. Differential Non-Linearity gives the user a measure of how much the input voltage may have changed in order to produce a one count change in the conversion result. Non-Linearity is the worst case deviation of code transitions from the corresponding code transitions of the Ideal Characteristic. Non-Linearity describes how much Differential Non-Linearities could add up to produce an overall maximum departure from a linear characteristic. If the Differential Non-Linearity errors are too large, it is possible for an A/D converter to miss codes or exhibit non-monotonicity. Neither behavior is desirable in a closed-loop system. A converter has no missed codes if there exists for each output code a unique input voltage range that produces that code only. A converter is monotonic if every subsequent code change represents an input voltage change in the same direction.

Differential Non-Linearity and Non-Linearity are quantified by measuring the Terminal Based Linearity Errors. A Terminal Based Characteristic results when an Actual Characteristic is shifted and rotated to eliminate zero offset and full-scale error (see Figure 17). The Terminal Based Characteristic is similar to the Actual Characteristic that would be seen if zero offset and full-scale error were externally trimmed away. In practice, this is done by using input circuits which include gain and offset trimming. In addition, VREF on the 8096BH could also be closely regulated and trimmed within the specified range to affect full-scale error.

Other factors that affect a real A/D Converter system include sensitivity to temperature, failure to completely reject all unwanted signals, multiplexer channel dissimilarities and random noise. Fortunately these effects are small.

Temperature sensitivities are described by the rate at which typical specifications change with a change in temperature.

Undesired signals come from three main sources. First, noise on VCC—VCC Rejection. Second, input signal

changes on the channel being converted after the sample window has closed—Feedthrough. Third, signals applied to channels not selected by the multiplexer—Off-Isolation.

Finally, multiplexer on-channel resistances differ slightly from one channel to the next causing Channel-to-Channel Matching errors, and random noise in general results in Repeatability errors.

3.5 8X9X A/D Converter Differences

The 8X9X A/D Converter does not have an internal Sample-and-Hold, and the conversion time is 168 state times (42 μ s with 12 MHz clock). These differences primarily influence the interface circuitry and the rate at which sampling can be done.

For the 8X9X, the idealized circuit in Figure 13 is still applicable. The only real difference is that the capacitor labeled C_S has a smaller value on 8X9X devices, but it is charged 10 times during a conversion. Since the actual C_S on 8X9X parts is about 0.5 pF, an effective C_S of 5.0 pF (10×0.5 pF) can be used as the internal capacitance that must be charged during a conversion. The value of R_I and I_L are nominally 5 k Ω and 3 μ A respectively.

Given these values, external circuits with source impedances of 1 k Ω or less will be able to maintain an input voltage within a tolerance of about ± 0.6 LSB (1.0 k $\Omega \times 3.0 \mu$ A = 3.0 mV) given the D.C. leakage. Source impedances above 2 k Ω will induce an external error of at least one LSB due to the voltage drop caused by the 3 μ A leakage. In addition, source impedances above 25 k Ω may degrade converter accuracy as a result of inadequate internal capacitor charging.

On 8X9X devices, the analog input is sampled 10 times while a conversion is in progress. Therefore, the input must remain stable so that conversion accuracy is not affected. If the input signal could vary significantly while a conversion is in progress, an external capacitor attached directly to the analog input pin could be used as a Sample-and-Hold. Since the internal capacitance is around 5.0 pF, an external 0.01 μ F capacitor (2048×5.0 pF) should provide an accurate input voltage to ± 0.5 LSB. If there is leakage on the capacitor, the value of the capacitor must be increased to compensate for the leakage. For example, assuming just the 3 μ A D.C. leakage caused by the 8X9X, 1 mV (less than 0.25 LSB) will be lost from a 0.15 μ F capacitor in 42 μ s. Therefore, the capacitor connected externally to the pin should be at least 0.2 μ F. However, if the external signal changes slowly relative to the conversion time (168 state times), it is recommended that the largest acceptable capacitance be used given the input signal frequency.

Figure 14 shows a simple interface which could be applicable to 8X9X devices if the size of the capacitor attached to the analog input pin is increased to a value greater than 0.2 μ F. The circuit in the figure also provides limited protection against over-voltage conditions on the analog inputs. However, before any circuit is used in an actual application, it should be thoroughly analyzed for applicability to the specific problem at hand.

3.6 A/D Glossary of Terms

Figures 15, 16 and 17 display many of these terms.

ABSOLUTE ERROR—The maximum difference between corresponding actual and ideal code transitions. Absolute Error accounts for all deviations of an actual converter from an ideal converter.

ACTUAL CHARACTERISTIC—The characteristic of an actual converter. The characteristic of a given converter may vary over temperature, supply voltage, and frequency conditions. An Actual Characteristic rarely has ideal first and last transition locations or ideal code widths. It may even vary over multiple conversion under the same conditions.

BREAK-BEFORE-MAKE—The property of a multiplexer which guarantees that a previously selected channel will be deselected before a new channel is selected. (e.g. the converter will not short inputs together.)

CHANNEL-TO-CHANNEL MATCHING—The difference between corresponding code transitions of actual characteristics taken from different channels under the same temperature, voltage and frequency conditions.

CHARACTERISTIC—A graph of input voltage versus the resultant output code for an A/D converter. It describes the transfer function of the A/D converter.

CODE—The digital value output by the converter.

CODE CENTER—The voltage corresponding to the midpoint between two adjacent code transitions.

CODE TRANSITION—The point at which the converter changes from an output code of Q , to a code of $Q + 1$. The input voltage corresponding to a code transition is defined to be that voltage which is equally likely to produce either of two adjacent codes.

CODE WIDTH—The voltage corresponding to the difference between two adjacent code transitions.

CROSSTALK—See "Off-Isolation".

D.C. INPUT LEAKAGE—Leakage current to ground from an analog input pin.

DIFFERENTIAL NON-LINEARITY—The difference between the ideal and actual code widths of the terminal based characteristic of a converter.

FEEDTHROUGH—Attenuation of a voltage applied on the selected channel of the A/D converter after the sample window closes.

FULL SCALE ERROR—The difference between the expected and actual input voltage corresponding to the full scale code transition.

IDEAL CHARACTERISTIC—A characteristic with its first code transition at $V_{IN} = 0.5 \text{ LSB}$, its last code transition at $V_{IN} = (V_{REF} - 1.5 \text{ LSB})$ and all code widths equal to one LSB.

INPUT RESISTANCE—The effective series resistance from the analog input pin to the sample capacitor.

LSB—LEAST SIGNIFICANT BIT: The voltage value corresponding to the full scale voltage divided by 2^n , where n is the number of bits of resolution of the converter. For a 10-bit converter with a reference voltage of 5.12 volts, one LSB is 5.0 mV. Note that this is different than digital LSBs, since an uncertainty of two LSB, when referring to an A/D converter, equals 10 mV. (This has been confused with an uncertainty of two digital bits, which would mean four counts, or 20 mV.)

MONOTONIC—The property of successive approximation converters which guarantees that increasing input voltages produce adjacent codes of increasing value, and that decreasing input voltages produce adjacent codes of decreasing value.

NO MISSED CODES—For each and every output code, there exists a unique input voltage range which produces that code only.

NON-LINEARITY—The maximum deviation of code transitions of the terminal based characteristic from the corresponding code transitions of the ideal characteristics.

OFF-ISOLATION—Attenuation of a voltage applied on a deselected channel of the A/D converter. (Also referred to as Crosstalk.)

REPEATABILITY—The difference between corresponding code transitions from different actual characteristics taken from the same converter on the same channel at the same temperature, voltage and frequency conditions.

RESOLUTION—The number of input voltage levels that the converter can unambiguously distinguish between. Also defines the number of useful bits of information which the converter can return.

SAMPLE DELAY—The delay from receiving the start conversion signal to when the sample window opens.

SAMPLE DELAY UNCERTAINTY—The variation in the Sample Delay.

SAMPLE TIME—The time that the sample window is open.

SAMPLE TIME UNCERTAINTY—The variation in the sample time.

SAMPLE WINDOW—Begins when the sample capacitor is attached to a selected channel and ends when the sample capacitor is disconnected from the selected channel.

SUCCESSIVE APPROXIMATION—An A/D conversion method which uses a binary search to arrive at the best digital representation of an analog input.

TEMPERATURE COEFFICIENTS—Change in the stated variable per degree centigrade temperature change. Temperature coefficients are added to the typical values of a specification to see the effect of temperature drift.

TERMINAL BASED CHARACTERISTIC—An Actual Characteristic which has been rotated and translated to remove zero offset and full-scale error.

VCC REJECTION—Attenuation of noise on the VCC line to the A/D converter.

ZERO OFFSET—The difference between the expected and actual input voltage corresponding to the first code transition.

4.0 ANALOG OUTPUTS

Analog outputs can be generated by two methods, either by using the PWM output or the HSO. Either device will generate a rectangular pulse train that varies in duty cycle and (for the HSO only) period. If a smooth analog signal is desired as an output, the rectangular waveform must be filtered.

In most cases this filtering is best done after the signal is buffered to make it swing from 0 to 5 volts since both of the outputs are guaranteed only to TTL levels. A block diagram of the type of circuit needed is shown in Figure 18. By proper selection of components, accounting for temperature and power supply drift, a highly accurate 8-bit D to A converter can be made using either the HSO or the PWM output. Figure 19 shows two typical circuits. If the HSO is used the accuracy could be theoretically extended to 16-bits, however the temperature and noise related problems would be extremely hard to handle.

When driving some circuits it may be desirable to use unfiltered Pulse Width Modulation. This is particularly true for motor drive circuits. The PWM output can be used to generate these waveforms if a fixed period on the order of 64 μ s is acceptable. If this is not the case then the HSO unit can be used. The HSO can generate a variable waveform with a duty cycle variable in up to 65536 steps and a period of up to 131 milliseconds. Both of these outputs produce TTL levels.

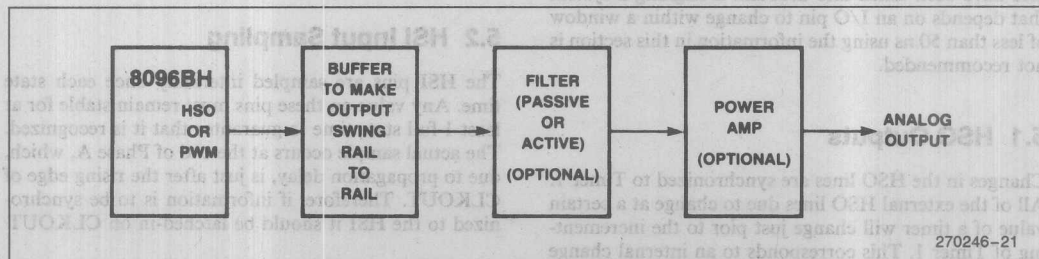


Figure 18. D/A Buffer Block Diagram

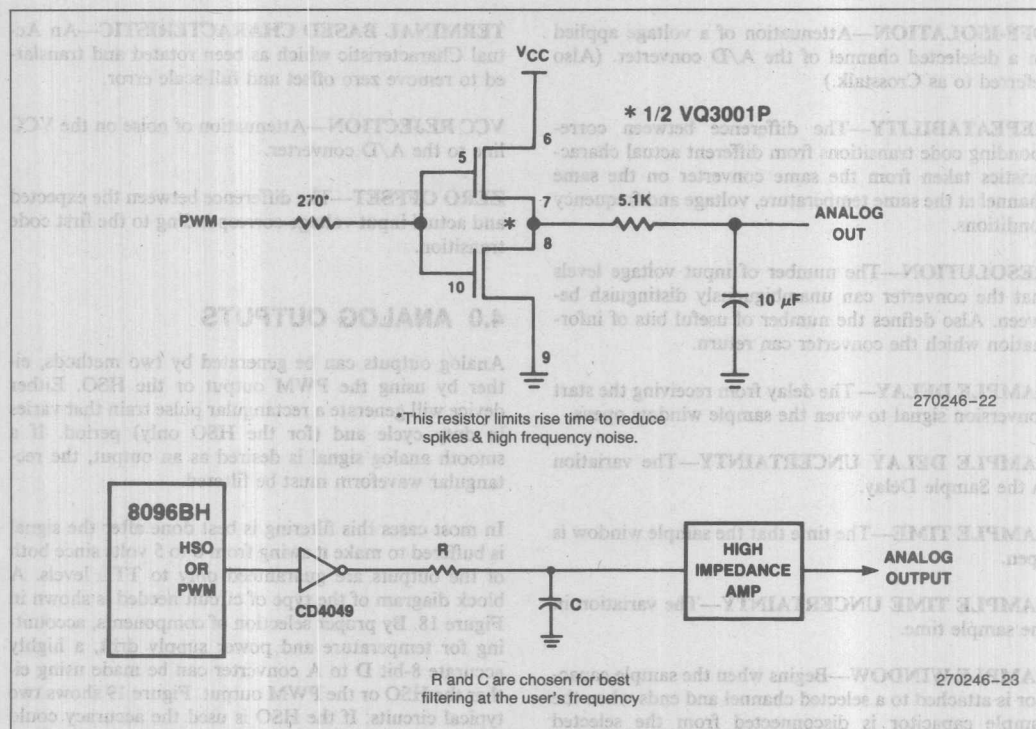


Figure 19. Buffer Circuits for D/A

5.0 I/O TIMINGS

The I/O pins on the 8096BH are sampled and changed at specific times within an instruction cycle. The changes occur relative to the internal phases shown in Figure 4. Note that the delay from XTAL1 to the internal clocks range from about 30 ns to 100 ns over process and temperature. Signals generated by internal phases are further delayed by 5 ns to 15 ns. The timings shown in this section are idealized; no propagation delay factors have been taken into account. Designing a system that depends on an I/O pin to change within a window of less than 50 ns using the information in this section is not recommended.

5.1 HSO Outputs

Changes in the HSO lines are synchronized to Timer 1. All of the external HSO lines due to change at a certain value of a timer will change just prior to the incrementing of Timer 1. This corresponds to an internal change

during Phase B every eight state times. From an external perspective the HSO pin should change just prior to the rising edge of CLKOUT and be stable by its falling edge. Information from the HSO can be latched on the CLKOUT falling edge. Internal events can occur anytime during the 8 state time window.

Timer 2 is synchronized to increment no faster than Timer 1, so there will always be at least one incrementing of Timer 1 while Timer 2 is at a specific value.

5.2 HSI Input Sampling

The HSI pins are sampled internally once each state time. Any value on these pins must remain stable for at least 1 full state time to guarantee that it is recognized. The actual sample occurs at the end of Phase A, which, due to propagation delay, is just after the rising edge of CLKOUT. Therefore, if information is to be synchronized to the HSI it should be latched-in on CLKOUT

falling. The time restriction applies even if the divide by eight mode is being used. If two events occur on the same pin within the same 8 state time window, only one of the events will be recorded. If the events occur on different pins they will always be recorded, regardless of the time difference. The 8 state time window, (i.e. the amount of time during which Timer 1 remains constant), is stable to within about 20 ns. The window starts roughly around the rising edge of CLKOUT, however this timing is very approximate due to the amount of internal circuitry involved.

5.3 Standard I/O Port Pins

Port 0 is different from the other digital ports in that it is actually part of the A/D converter. The port is sampled once every state time, however, sampling is not synchronized to Timer 1. If this port is used, the input signal on the pin must be stable one state time before the reading of the SFR.

On 8X9X devices, Port 0 is sampled every eight state times (the same frequency at which the comparator is charged-up during an A/D conversion). This 8 state time counter is not synchronized with Timer 1. If this port is used, the input signal on the pin must be stable 8 state times prior to reading the SFR.

Port 1 and Port 2 have quasi-bidirectional I/O pins. When used as inputs the data on these pins must be stable one state time prior to reading the SFR. This timing is also valid for the input-only pins of Port 2 and is similar to the HSI in that the sample occurs just after the rising edge of CLKOUT. When used as outputs, the quasi-bidirectional pins will change state shortly after CLKOUT falls. If the change was from '0' to a '1' the low impedance pullup will remain on for one state time after the change.

Ports 3 and 4 are addressed as off-chip memory-mapped I/O. The port pins will change state shortly after the rising edge of CLKOUT. When these pins are used as Ports 3 and 4 they are open drains, their structure is different when they are used as part of the bus. See Section 10.4 of the MCS-96 Architecture chapter. Additional information on port reconstruction is available in Section 7.8 of this chapter.

6.0 SERIAL PORT TIMINGS

The serial port on the 8096BH was designed to be compatible with the 8051 serial port. Since the 8051 uses a divide by 2 clock and the 8096BH uses a divide by 3, the serial port on the 8096BH had to be provided with its own clock circuit to maximize its compatibility with

the 8051 at high baud rates. This means that the serial port itself does not know about state times. There is circuitry which is synchronized to the serial port and to the rest of the 8096BH so that information can be passed back and forth.

The baud rate generator is clocked by either XTAL1 or T2CLK. Because T2CLK needs to be synchronized to the XTAL1 signal its speed must be limited to $\frac{1}{16}$ that of XTAL1. The serial port will not function during the time between the consecutive writes to the baud rate register. Section 11.4 of the MCS-96 Architecture chapter discusses programming the baud rate generator.

6.1 Mode 0

Mode 0 is the shift register mode. The TXD pin sends out a clock train, while the RXD pin transmits or receives the data. Figure 20 shows the waveforms and timing. Note that the port starts functioning when a '1' is written to the REN (Receiver Enable) bit in the serial port control register. If REN is already high, clearing the RI flag will start a reception.

In this mode the serial port can be used to expand the I/O capability of the 8096BH by simply adding shift registers. A schematic of a typical circuit is shown in Figure 21. This circuit inverts the data coming in, so it must be reinverted in software. The enable and latch connections to the shift registers can be driven by decoders, rather than directly from the low speed I/O ports, if the software and hardware are properly designed.

6.2 Mode 1 Timings

Mode 1 operation of the serial port makes use of 10-bit data packages, a start bit, 8 data bits and a stop bit. The transmit and receive functions are controlled by separate shift clocks. The transmit shift clock starts when the baud rate generator is initialized, the receive shift clock is reset when a '1 to 0' transition (start bit) is received. The transmit clock may therefore not be in sync with the receive clock, although they will both be at the same frequency.

The TI (Transmit Interrupt) and RI (Receive Interrupt) flags are set to indicate when operations are complete. TI is set when the last data bit of the message has been sent, not when the stop bit is sent. If an attempt to send another byte is made before the stop bit is sent the port will hold off transmission until the stop bit is complete. RI is set when 8 data bits are received, not when the stop bit is received. Note that when the serial port status register is read both TI and RI are cleared.

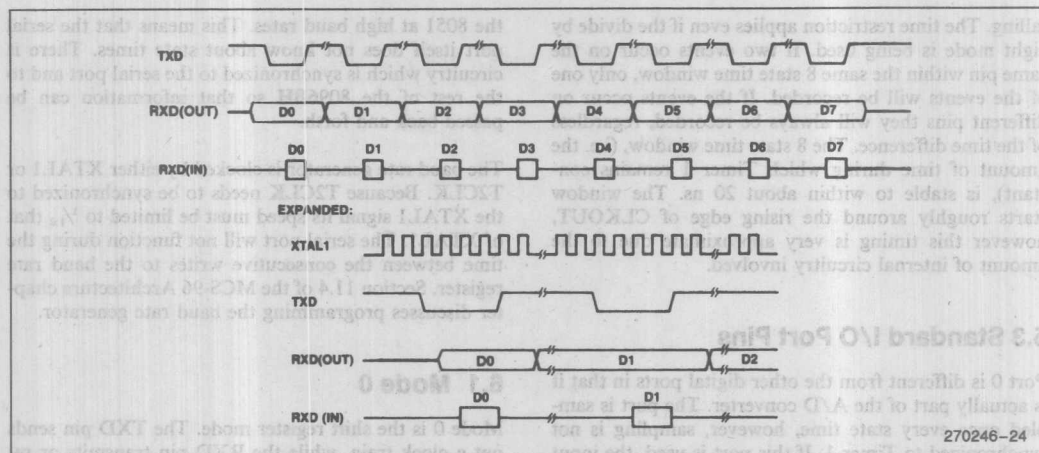


Figure 20. Serial Port Timings in Mode 0

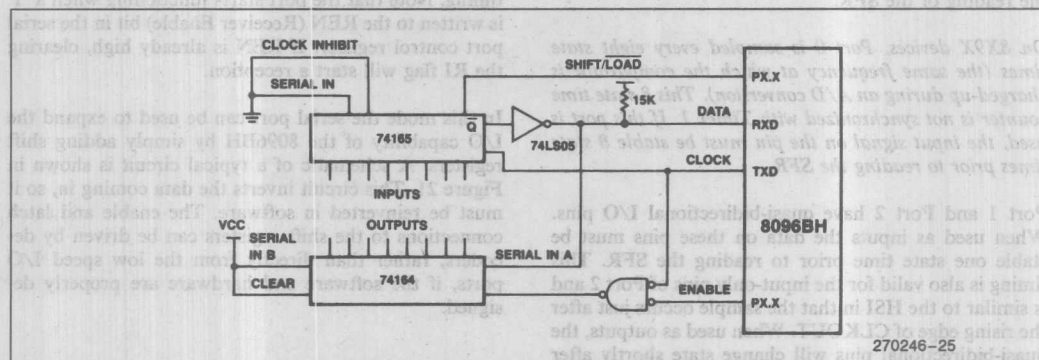


Figure 21. Mode 0 Serial Port Example

Caution should be used when using the serial port to connect more than two devices in half-duplex, (i.e. one wire for transmit and receive). If the receiving processor does not wait for one bit time after RI is set before starting to transmit, the stop bit on the link could be squashed. This could cause a problem for other devices listening on the link.

6.3 Mode 2 and 3 Timings

Modes 2 and 3 operate in a manner similar to that of Mode 1. The only difference is that the data is now made up of 9 bits, so 11-bit packages are transmitted and received. This means that TI and RI will be set on the 9th data bit rather than the 8th. The 9th bit can be used for parity or multiple processor communications (see Section 11 of the MCS-96 Architecture chapter).

7.0 BUS TIMING AND MEMORY INTERFACE

7.1 Bus Functionality

The 8096BH has a multiplexed (address/data) bus which can be dynamically configured to have an 8-bit or 16-bit data width. There are control lines to demultiplex the bus (ALE or ADV), indicate reads (RD), indicate writes (WRL and WRH, or WR with BHE and AD0), and a signal to indicate accesses that are for an instruction fetch (INST). Section 3.5 of the MCS-96 Architecture chapter contains an overview of the bus operation.

On 8X9X devices only the 16-bit multiplexed bus is available. In addition, on 8X9X devices the WRL and WRH signals are not available and the functionality of the BHE and INST lines differs from the 8X9XBH devices. See the data sheet of the device that you use.

number of inserted wait states is equal to the limit set in the Chip Configuration Register (see Section 2 of the MCS-96 Architecture chapter). There is a maximum time that the **READY** line can be held low without risking a processor malfunction due to dynamic nodes that have not been refreshed during the wait states. This time is shown as **TYLYH** in the data sheet.

In most cases the **READY** line is brought low after the address is decoded and it is determined that a wait state is needed. It is very likely that some addresses, such as those addressing memory mapped peripherals, would need wait states, and others would not. The **READY** line must be stable within the **TLLYV** specification after **ALE** falls or the processor could lock-up. There is no requirement as to when **READY** may go high, as long as the maximum **READY** low time (**TYLYH**) is not violated. To ensure that only one wait state is inserted it is necessary to provide external circuitry which brings **READY** high **TLLYH** after the falling edge of **ALE/ADV**, or program the Chip Configuration Register to select a Ready Control limit of one.

Internally, the chip latches **READY** on the first falling edge of Phase A after **ALE/ADV** falls. Phase A is buffered and brought out externally as **CLOCKOUT**, so **CLOCKOUT** is a delayed Phase A. If a 1 is seen, the bus cycle proceeds uninterrupted with no wait state insertions. If a 0 is seen, one wait state (3 T_{osc}) is inserted.

If a wait state is inserted, **READY** is internally latched on the next rising edge of Phase A. If a 1 is found the bus cycle resumes with the net impact being the insertion of one wait state. If a 0 is seen, a second wait state is inserted.

The **READY** pin is again latched on the next rising edge of **CLOCKOUT** if two wait states were inserted. If the chip sees a 1, the bus cycle is resumed with the result being an insertion of two wait states. If another 0 is seen, a third wait state is inserted in the bus cycle and

Definitions of A.C. timing specifications differ slightly on 8X9X devices. See the data sheet for the part you are using for more information.

T_{osc}—Oscillator Period, one cycle time on XTAL1.

Timings the Memory System Must Meet

TLLYH—**ALE/ADV** low to **READY** high: Maximum time after **ALE/ADV** falls until **READY** is brought high to ensure no more wait states. If this time is exceeded unexpected wait states may result. Nominally 1 T_{osc} + 3 T_{osc} × number of wait states desired.

TLLYV—**ALE/ADV** low to **READY** low: Maximum time after **ALE/ADV** falls until **READY** must be valid. If this time is exceeded the part could malfunction necessitating a chip reset. Nominally 2 T_{osc} periods.

TCLYX—**READY** hold after **CLOCKOUT** low: Minimum time that the value on the **READY** pin must be valid after **CLOCKOUT** falls. The minimum hold time is always zero nanoseconds.

TYLYH—**READY** low to **READY** high: Maximum time the part can be in the not-ready state. If it is exceeded, the 8096BH dynamic nodes which hold the current instruction may 'forget' how to finish the instruction.

TAVDV—**ADDRESS** valid to **DATA** valid: Maximum time that the memory has to output valid data

after the 8096BH outputs a valid address. Nominally, a maximum of 5 T_{osc} periods.

TAVGV—**ADDRESS** valid to **BUSWIDTH** valid: Maximum time after **ADDRESS** becomes valid until **BUSWIDTH** must be valid. Nominally less than 2 T_{osc} periods.

TLLGV—**ALE/ADV** low to **BUSWIDTH** valid: Maximum time after **ALE/ADV** is low until **BUSWIDTH** must be valid. If this time is exceeded the part could malfunction necessitating a chip reset. Nominally less than 1 T_{osc}.

TLLGX—**BUSWIDTH** hold after **ALE/ADV** low: Minimum time that **BUSWIDTH** must be valid after **ALE/ADV** is low. Nominally 1 T_{osc}.

TRLDV—**READ** low to **DATA** valid: Maximum time that the memory has to output data after **READ** goes low. Nominally, a maximum of 3 T_{osc} periods.

TRHDZ—**READ** high to **DATA** float: Time after **READ** is high until the memory must float the bus. The memory signal can be removed as soon as **READ** is not low, and must be removed within the specified maximum time from when **READ** is high. Nominally a maximum of 1 T_{osc} period.

TRHDX—**DATA** hold after **READ** goes high: Minimum time that memory must hold input **DATA** valid after **RD** is high. The hold time minimum is always zero nanoseconds.

Figure 23. Timing Specification Explanations

Timings the 8096 Will Provide

TOHCH—XTAL1 high to **CLOCKOUT** high: Delay from the rising edge of XTAL1 to the resultant rising edge on **CLOCKOUT**. Needed in systems where the signal driving XTAL1 is also used as a clock for external devices. Typically 50 to 100 nanoseconds.

TCHCH—**CLKOUT** high to **CLKOUT** high: The period of **CLKOUT** and the duration of one state time. Always 3 Tosc average, but individual periods could vary by a few nanoseconds.

TCHCL—**CLKOUT** high to **CLKOUT** low: Nominally 1 Tosc period.

TCLLH—**CLKOUT** low to **ALE** high: A help in deriving other timings. Typically plus or minus 5 ns to 10 ns.

TCLVL—**CLOCKOUT** low to **ALE/ADV** low: A help in deriving other timings. Nominally 1 Tosc.

TLLCH—**ALE/ADV** low to **CLKOUT** high: Used to derive other timings, nominally 1 Tosc period.

TLHLL—**ALE/ADV** high to **ALE/ADV** low: **ALE/ADV** high time. Useful in determining **ALE/ADV** rising edge to **ADDRESS** valid time. Nominally 1 Tosc period for **ALE** and 1 Tosc for **ADV** with back-to-back bus cycles.

TAVLL—**ADDRESS** valid to **ALE/ADV** low: Length of time **ADDRESS** is valid before **ALE/ADV** falls. Important timing for address latch circuitry. Nominally 1 Tosc period.

TLLAX—**ALE/ADV** low to **ADDRESS** invalid: Length of time **ADDRESS** is valid after **ALE/ADV** falls. Important timing for address latch circuitry. Nominally 1 Tosc period.

TLLRL—**ALE/ADV** low to **READ** or **WRITE** low: Length of time after **ALE/ADV** falls before **RD** or **WR** fall. Could be needed to ensure that proper memory decoding takes place before it is output enabled. Nominally 1 Tosc period.

TLLHL—**ALE/ADV** low to **WRL**, **WRH** low: Minimum time after **ALE/ADV** is low that the write strobe signals will go low. Could be needed to ensure

that proper memory decoding takes place before it is output enabled. Nominally 2 Tosc periods.

TRLRH—**READ** low to **READ** high: **RD** pulse width, nominally 1 Tosc period.

TRHLH—**READ** high to **ALE/ADV** high: Time between **RD** going inactive and next **ALE/ADV**, also used to calculate time between **RD** inactive and next **ADDRESS** valid. Nominally 1 Tosc period.

TRHBX—**READ** high to **INST**, **BHE**, **AD8-15** Inactive: Minimum time that the **INST** and **BHE** lines will be valid after **RD** goes high. Also the minimum time that the upper eight address lines (8-bit bus mode) will remain valid after **RD** goes high. Nominally 1 Tosc.

TWHBX—**WRITE** high to **INST**, **BHE**, **AD8-15** Inactive: Minimum time that the **INST** and **BHE** lines will be valid after **WR** goes high. Also the minimum time that the upper eight address lines (8-bit bus mode) will remain valid after **WR** goes high. Nominally 1 Tosc.

TWLWH—**WRITE** low to **WRITE** high: Write pulse width, nominally 3 Tosc periods.

THLHH—**WRL**, **WRH** low to **WRL**, **WRH** high: Write strobe signal pulse width. Nominally 2 Tosc periods.

TQVHL—**OUTPUT** valid to **WRL**, **WRH** low: Minimum time that **OUTPUT** data is valid prior to write strobes becoming active. Needed for interfacing to memories that read data on the falling edge of write. Nominally 1 Tosc.

TQVWH—**OUTPUT** valid to **WRITE** high: Time that the **OUTPUT** data is valid before **WR** is high. Nominally 3 Tosc periods.

TWHQX—**WRITE** high to **OUTPUT** not valid: Time that the **OUTPUT** data is valid after **WR** is high. Nominally 1 Tosc period.

TWHLH—**WRITE** high to **ALE/ADV** high: Time between write high and next **ALE/ADV**, also used to calculate the time between **WR** high and next **ADDRESS** valid. Nominally 2 Tosc periods.

Figure 23. Timing Specification Explanations (Continued)

the **READY** pin is again latched on the following rising edge of **CLOCKOUT**. If internal Ready Control is not used, the **READY** line must at this point be a 1 to ensure proper operation.

On 8X9X devices there is no internal Ready Control, therefore, external circuitry must completely control the insertion of wait states into 8X9X bus cycles.

7.4 INST Line Usage

The INST (Instruction) line is high during bus cycles that are for an instruction fetch and low for any other bus cycle. The INST signal (not present on 48-pin versions) can be used with a logic analyzer to debug a system. In this way it is possible to determine if a fetch was for instructions or data, making the task of tracing the program much easier.

On 8X9X devices the INST line is high during the output of an address that is for an instruction fetch. It is low during the same time for any other memory access. At any other time it is not valid.

7.5 BUSWIDTH Pin Usage

The BUSWIDTH pin is a control input which determines the width of the bus access in progress. BUSWIDTH is sampled after the rising edge of the first CLOCKOUT after ALE/ADV goes low. If a one is seen, the bus access progresses as a 16-bit cycle. If a zero is seen, the bus access progresses as an 8-bit cycle. The BUSWIDTH setup and hold timing requirements appear in the data sheet.

The BUSWIDTH pin can be overridden by causing the BUS WIDTH SELECT bit in the Chip Configuration Register (CCR) to be zero. This will permanently select an 8-bit bus width. However, if the BUS WIDTH SELECT bit in the CCR is a one, the BUSWIDTH pin determines the bus width. See Section 3.5 of the MCS-96 Architecture chapter. Since the BUSWIDTH pin is not available on 48-pin parts, the BUS WIDTH SELECT bit in the CCR determines bus width.

On 8X9X devices, the 8-bit bus is not available, the CCR does not exist and the BUSWIDTH pin is named the TEST pin. The TEST pin is used for testing purposes and should be tied to VCC in application circuits.

7.6 Address Decoding

The multiplexed bus of the 8096BH must be demultiplexed before it can be used. This can be done with two 74LS373 transparent latches for an 8096BH in 16-bit bus mode, or one 74LS373 for an 8096BH in 8-bit bus mode. As explained in Section 3.5 of the MCS-96 Architecture chapter, the latched address signals will be referred to as MA0 through MA15 (Memory Address), and the data lines will be called MD0 through MD15 (Memory Data).

Since the 8096BH can make accesses to memory for either bytes or words, it is necessary to have a way of determining the type of access desired when the bus is 16-bits wide. For write cycles, the signals Write Low (WRL) and Write High (WRH) are provided. WRL will go low during all word writes and during all byte writes to an even location. Similarly, WRH will go low during all word writes and during all byte writes to an odd location. During read cycles, an 8096BH in 16-bit bus mode will always do a word read of an even location. If only one byte of the word is needed, the chip discards the byte it does not need.

Since 8096BH memory accesses over an 8-bit wide bus are always bytes, only one write strobe is needed for write cycles. For this purpose the WRL signal was made to go low for all write cycles during 8-bit bus accesses. When a word operation is requested, the bus controller performs two byte-wide bus cycles.

In many cases it may be desirable to have a write signal with a longer pulse width than WRL/WRH. The Write (WR) line of the 8096BH is an alternate control signal that shares a pin with WRL and is only available in 16-bit bus mode. WR is nominally one TOSC longer than the WRL/WRH signals, but goes low for any write cycle. Therefore it is necessary to decode for the type of write (byte or word) desired.

The Byte High Enable (BHE) signal and MA0 can be used for this purpose. BHE is an alternate control sig-

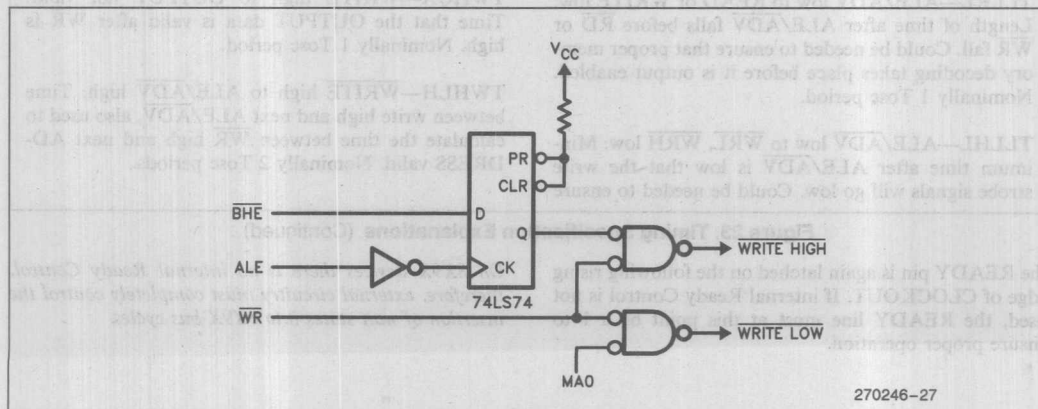


Figure 24. Decoding WR and BHE to Generate WriteLow and WriteHigh

nal that shares a pin with \overline{WRH} . When \overline{BHE} is low, the high byte of the 16-bit bus is enabled. When $\overline{MA0}$ is low, the lower byte is enabled. When $\overline{MA0}$ is low and \overline{BHE} is low, both bytes are enabled. Figure 24 shows how to use \overline{WR} , \overline{BHE} and $\overline{MA0}$ to decode bus accesses. It's important to note that this decoding inserts a delay in the write signal which must be considered in a system timing analysis.

On 8X9X devices, only the \overline{RD} , \overline{WR} and \overline{BHE} signals are available for bus control. This means that discriminating between byte and word bus accesses must be done by decoding \overline{WR} , \overline{BHE} and $\overline{MA0}$ as described above.

Further, the \overline{WR} signal on 8X9X devices is nominally the same width as the \overline{WRL} and \overline{WRH} signals. 8X9XBH devices (2 Tost), and the \overline{BHE} signal must be latched since it is valid only while the address is valid. See Figure 24 and the data sheet of the device that you use.

External memory systems for the 8096BH can be set up in many ways. Figures 25 through 28 show block diagrams of memory systems using an 8-bit bus with a single EPROM, using an 8-bit bus with RAM and EPROM, using a 16-bit bus with two external EPROMs and using a 16-bit bus in a RAM and ROM system.

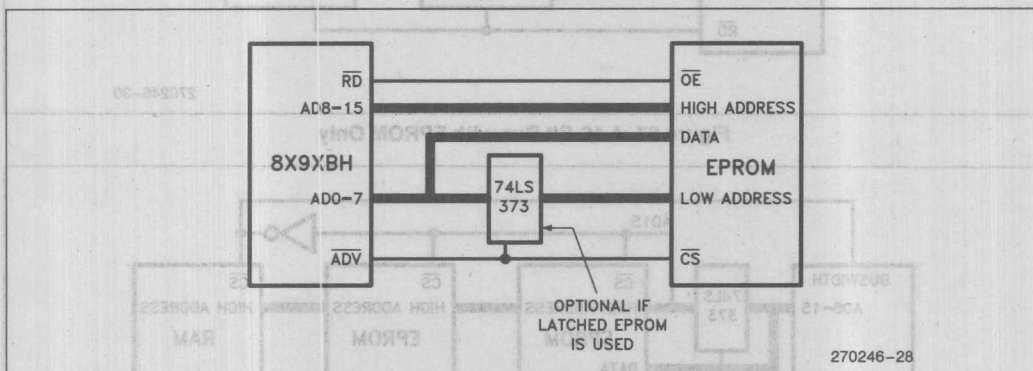


Figure 25. An 8-Bit Bus with EPROM Only

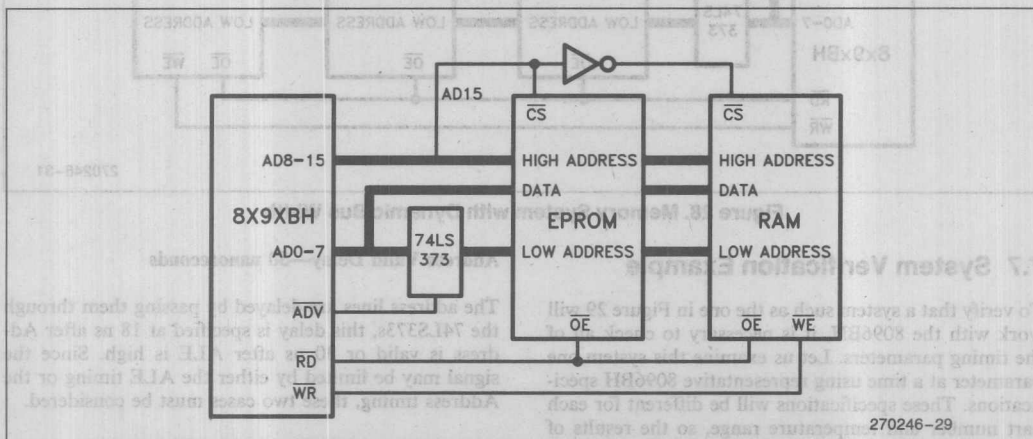


Figure 26. An 8-Bit Bus with EPROM and RAM

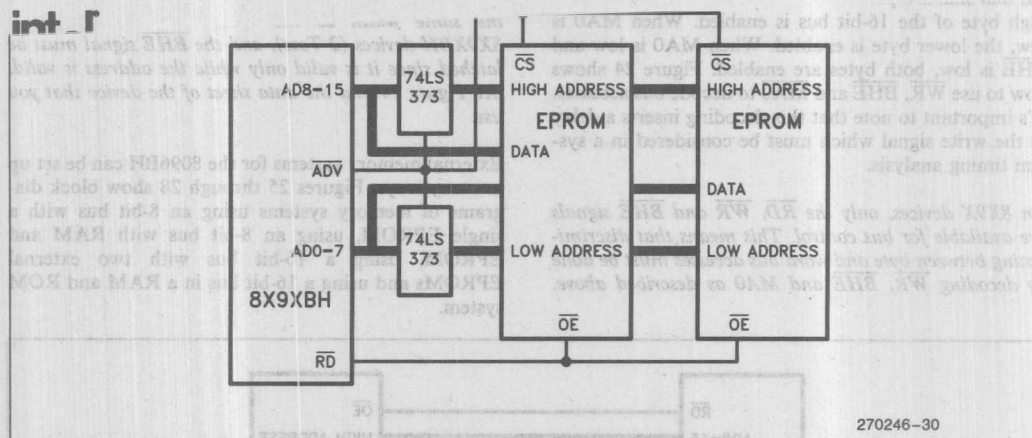


Figure 27. A 16-Bit Bus with EPROM Only

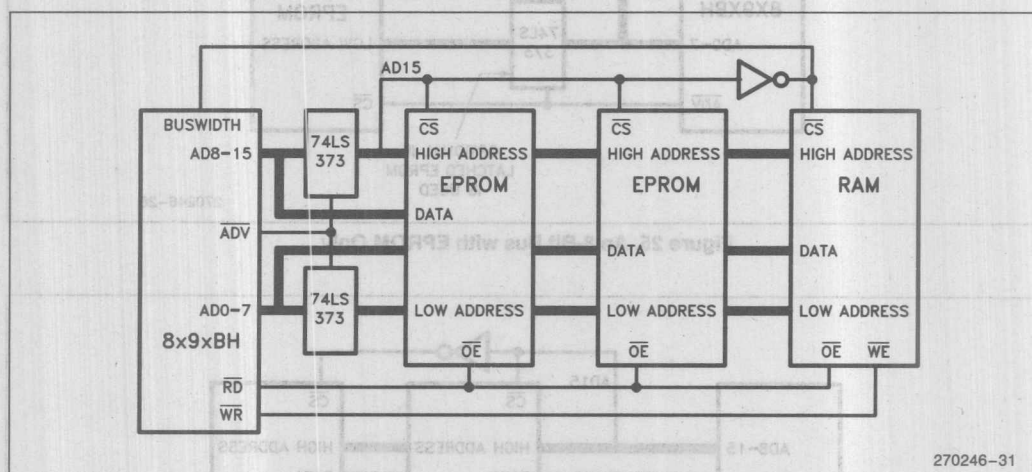


Figure 28. Memory System with Dynamic Bus Width

7.7 System Verification Example

To verify that a system such as the one in Figure 29 will work with the 8096BH, it is necessary to check all of the timing parameters. Let us examine this system one parameter at a time using representative 8096BH specifications. These specifications will be different for each part number and temperature range, so the results of this example must be modified based on the most recent data sheet for the specific part to be used.

The timings of signals that the processor and memory use are affected by the latch and buffer circuitry. The timings of the signal provided by the processor are delayed by various amounts of time. Similarly, the signals coming back from the memory are also delayed. The calculations involved in verifying this system follow:

Address Valid Delay—30 nanoseconds

The address lines are delayed by passing them through the 74LS373s, this delay is specified at 18 ns after Address is valid or 30 ns after ALE is high. Since the signal may be limited by either the ALE timing or the Address timing, these two cases must be considered.

If Limited by ALE

$$\text{Minimum ALE pulse width} = T_{\text{osc}} - 25 \text{ (TLHLL)}$$

$$\text{Minimum Addr set-up to ALE falling} = T_{\text{osc}} - 25 \text{ (TAVLL)}$$

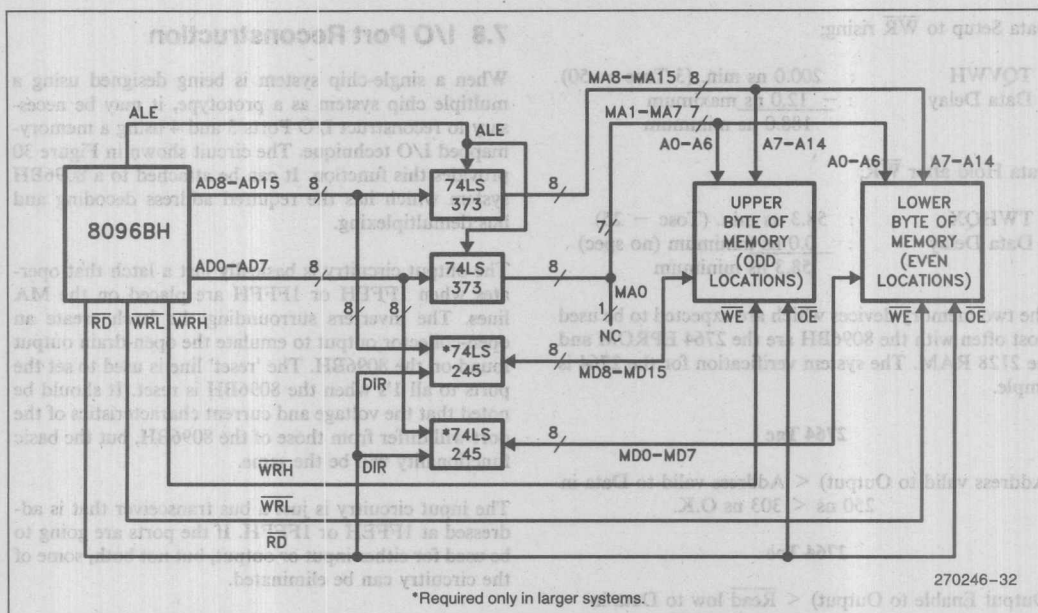


Figure 29. RAM/ROM Memory System

Therefore, in the worst case, ALE would occur 0 ns before Address valid.

Total delay from 8096BH Address stable to MA (Memory Address) stable would be:

$$\begin{array}{r} \text{ALE delay from address} \quad - \quad 0 \\ \text{74LS373 clock to output} \quad \underline{30} \\ \hline 30 \text{ nanoseconds} \end{array}$$

If Limited by Address Valid

74LS373 Data Valid to Data Output = 18 nanoseconds

In the worst case, the delay in Address valid is controlled by ALE and has a value of 30 nanoseconds.

Delay of Data Transfer to/from Processor—12 nano-seconds

The $\overline{\text{RD}}$ low to Data valid specification (TRLDV) is 3 Tasc - 50, (200 ns at 12 MHz). The 74LS245 is enabled by $\overline{\text{RD}}$ and has a delay of 40 ns from enable. The enable delay is clearly not a problem.

The 74LS245 is enabled for write, except during a read, so there is no enable delay to consider for write operations.

The Data In to Data Out delay of the 74LS245 is 12 ns.

CHARACTERISTICS OF A 12 MHz 8096BH SYSTEM WITH LATCHES

Required by system:

Address valid to Data in:

TAVDV	: 345.6 ns max. (5 Tosc - 70)
Address Delay	: - 30.0 ns maximum
Data Delay	: - 12.0 ns maximum
	303.6 ns maximum

Read low to Data in:

TRLDV : 200.0 ns max. (3 T_{osc} - 50)
 Address Delay : - 00.0 ns maximum
 Data Delay : - 12.0 ns maximum
 188.0 ns maximum

Provided by system:

Address valid to Control:

TLLRL	:	63.3 ns min. (Tosc - 20)
TAVLL	:	158.3 ns min. (Tosc - 25)
Address Delay	:	30.0 ns maximum
WR Delay	:	00.0 ns minimum
		91.6 ns minimum

Write Pulse Width:

THLHH : $\frac{146.6 \text{ ns}}{146.6 \text{ ns minimum}}$ min. (2 Tosc - 20)

Data Setup to \overline{WR} rising;

TQVWH : 200.0 ns min. (3 T_{osc} - 50)
Data Delay : - 12.0 ns maximum
188.0 ns minimum

Data Hold after \overline{WR} ;

TWHQX : 58.3 ns min. (T_{osc} - 25)
Data Delay : 0.0 ns minimum (no spec)
58.3 ns minimum

The two memory devices which are expected to be used most often with the 8096BH are the 2764 EPROM and the 2128 RAM. The system verification for the 2764 is simple.

2764 Tac

(Address valid to Output) < Address valid to Data in
250 ns < 303 ns O.K.

2764 Toe

(Output Enable to Output) < \overline{Read} low to Data in
100 ns < 188 ns O.K.

These calculations assume no address decoder delays and no delays on the \overline{RD} (OE) line. If there are delays in these signals the delays must be added to the 2764's timing.

The read calculations for the 2128 are similar to those for the 2764.

2128-20 Tac < Address valid to Data in
200 ns < 303 ns O.K.

2128-20 Toe < \overline{Read} low to Data in
65 ns < 188 ns O.K.

The write calculation are a little more involved, but still straight-forward.

2128 Twp (Write Pulse) < Write Pulse Width
100 ns < 146 ns O.K.

2128 Tds (Data Setup) < Data Setup to \overline{WR} rising
65 ns < 188 ns O.K.

2128 Tdh (Data Hold) < Data Hold after \overline{WR}
0 ns < 58 ns

All of the above calculations have been done assuming that no components are in the circuit except for those shown in Figure 29. If additional components are added, as may be needed for address decoding or memory bank switching, the calculations must be updated to reflect the actual circuit.

7.8 I/O Port Reconstruction

When a single-chip system is being designed using a multiple chip system as a prototype, it may be necessary to reconstruct I/O Ports 3 and 4 using a memory-mapped I/O technique. The circuit shown in Figure 30 provides this function. It can be attached to a 8096BH system which has the required address decoding and bus demultiplexing.

The output circuitry is basically just a latch that operates when 1FFEh or 1FFFh are placed on the MA lines. The inverters surrounding the latch create an open-collector output to emulate the open-drain output found on the 8096BH. The 'reset' line is used to set the ports to all 1's when the 8096BH is reset. It should be noted that the voltage and current characteristics of the port will differ from those of the 8096BH, but the basic functionality will be the same.

The input circuitry is just a bus transceiver that is addressed at 1FFEh or 1FFFh. If the ports are going to be used for either input or output, but not both, some of the circuitry can be eliminated.

8.0 NOISE PROTECTION TIPS

Designing controllers differs from designing other computer equipment in the area of noise protection. A microcontroller circuit under the hood of a car, in a photocopier, CRT terminal, or a high speed printer is subject to many types of electrical noise. Noise can get to the processor directly through the power supply, or it can be induced onto the board by electromagnetic fields. It is also possible for the PC board to find itself in the path of electrostatic discharges. Glitches and noise on the PC board can cause the processor to act unpredictably, usually by changing either the memory locations or the program counter.

There are both hardware and software solutions to noise problems, but the best solution is good design practice and a few ounces of prevention. The 8096BH has a Watchdog Timer which will reset the part if it fails to execute the software properly. The software should be set up to take advantage of this feature.

It is also recommended that unused areas of code be filled with NOPs and periodic jumps to an error routine or RST (reset chip) instructions. This is particularly important in the code around lookup tables, since if lookup tables are executed all sorts of bad things can happen. Wherever space allows, each table should be surrounded by 7 NOPs (the longest 8096BH instruction has 7 bytes) and a RST or jump to error routine instruction. This will help to ensure a speedy recovery should the processor have a glitch in the program flow.

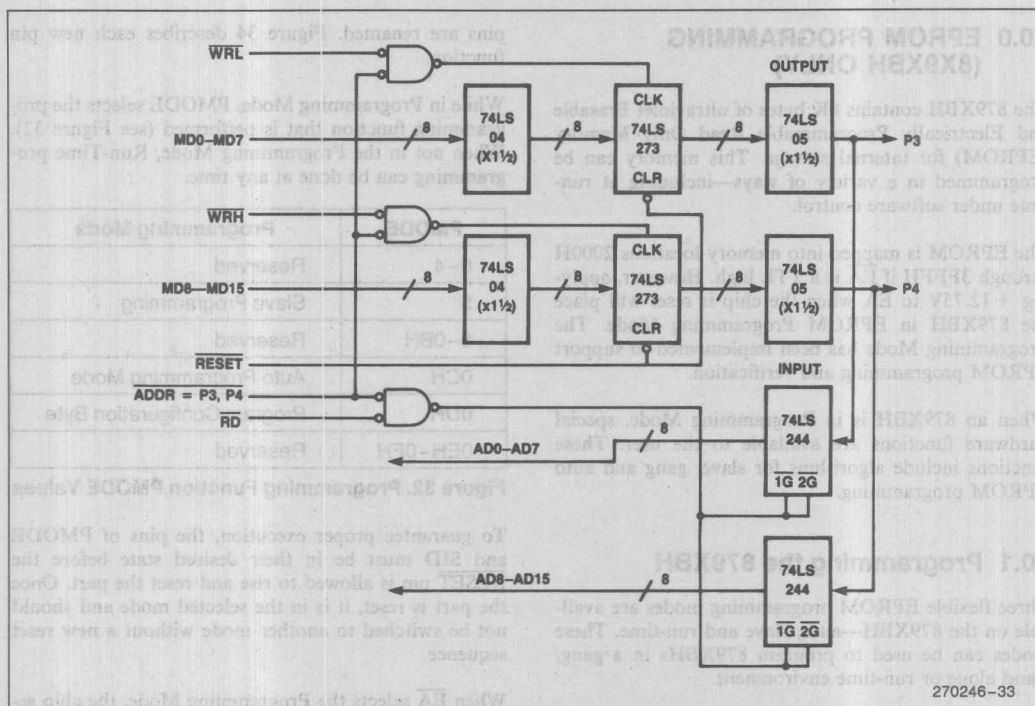


Figure 30. I/O Port Reconstruction

Many hardware solutions exist for keeping PC board noise to a minimum. Ground planes, gridded ground and VCC structures, bypass capacitors, transient absorbers and power busses with built-in capacitors can all be of great help. It is much easier to design a board with these features than to try to retrofit them later. Proper PC board layout is probably the single most important and, unfortunately, least understood aspect of project design. Minimizing loop areas and inductance, as well as providing clean grounds are very important. More information on protecting against noise can be found in the Application Note AP-125, "Designing Microcontroller Systems for Noisy Environments".

9.0 PACKAGING

The MCS-96 family of products is offered in many versions. They are available in 48-pin or 68-pin packages,

with or without on-chip ROM/EPROM and with or without an A/D converter. A summary of the available options is shown in Figure 31.

The 48-pin versions are available in ceramic and plastic 48-pin Dual-In-Line package (DIP). The ceramic versions have part numbers with the prefix "C". The plastic versions have the prefix "P".

The 68-pin versions are available in a ceramic pin grid array (PGA), a plastic leaded chip carrier (PLCC) and a Type B leadless chip carrier (LCC). PGA devices have part numbers with the prefix "C". PLCC devices have the prefix "N". LCC devices have the prefix "R".

Specifications for the various members of the MCS-96 family are contained in the next chapter.

	ROMless		With ROM		With EPROM	
	68-pin	48-pin	68-pin	48-pin	68-pin	48-pin
Without A to D	8096		8396		8796	
With A to D	8097	8095	8397	8395	8797	8795

Figure 31. The MCS[®]-96 Family of Products

10.0 EPROM PROGRAMMING (8X9XBH ONLY)

The 879XBH contains 8K bytes of ultraviolet Erasable and Electrically Programmable Read Only Memory (EPROM) for internal storage. This memory can be programmed in a variety of ways—including at run-time under software control.

The EPROM is mapped into memory locations 2000H through 3FFFFH if EA is a TTL high. However, applying +12.75V to EA when the chip is reset will place the 879XBH in EPROM Programming Mode. The Programming Mode has been implemented to support EPROM programming and verification.

When an 879XBH is in Programming Mode, special hardware functions are available to the user. These functions include algorithms for slave, gang and auto EPROM programming.

10.1 Programming the 879XBH

Three flexible EPROM programming modes are available on the 879XBH—auto, slave and run-time. These modes can be used to program 879XBHs in a gang, stand alone or run-time environment.

The Auto Programming Mode enables an 879XBH to program itself, and up to 15 other 879XBHs, with the 8K bytes of code beginning at address 4000H on its external bus. The Slave Mode provides a standard interface that enables any number of 879XBHs to be programmed by a master device such as an EPROM programmer. The Run-Time Mode allows individual EPROM locations to be programmed at run-time under complete software control.

In the Programming Mode, some I/O pins have been renamed. These new pin functions are used to determine the programming function that is performed, provide programming ALEs, provide slave ID numbers and pass error information. Figure 33 shows how the

pins are renamed. Figure 34 describes each new pin function.

While in Programming Mode, PMODE selects the programming function that is performed (see Figure 32). When not in the Programming Mode, Run-Time programming can be done at any time.

PMODE	Programming Mode
0-4	Reserved
5	Slave Programming
6-0BH	Reserved
0CH	Auto Programming Mode
0DH	Program Configuration Byte
0EH-0FH	Reserved

Figure 32. Programming Function PMODE Values

To guarantee proper execution, the pins of PMODE and SID must be in their desired state before the RESET pin is allowed to rise and reset the part. Once the part is reset, it is in the selected mode and should not be switched to another mode without a new reset sequence.

When \overline{EA} selects the Programming Mode, the chip reset sequence loads the CCR from the Programming Chip Configuration Byte (PCCB). This is a separate EPROM location that is not mapped under normal operation. PCCB is only important when programming in the Auto Programming Mode. In this mode, the 879XBH that is being programmed gets the data to be programmed from external memory over the system bus. Therefore, PCCR must correctly correspond to the memory system in the programming setup, which is not necessarily the memory organization of the application.

The following sections describe 879XBH programming in each programming mode.

Without A to D	ROMless		With ROM		With EPROM	
	88-pin	48-pin	88-pin	48-pin	88-pin	48-pin
Without A to D	8088	8088	8088	8088	8788	8788
With A to D	8087	8088	8087	8088	8787	8788

Figure 34. The MCS-96 Family of Products

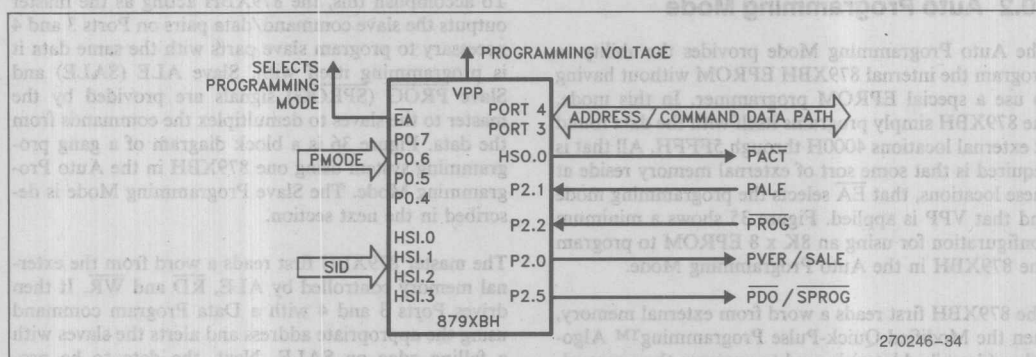


Figure 33. Programming Mode Pin Function

Name	Function
PMODE	PROGRAMMING MODE SELECT: Determines the EPROM programming algorithm that is performed. PMODE is sampled after a chip reset and should be static while the part is operating.
SID	SLAVE ID NUMBER: Used to assign each slave pin of Port 3 or 4 to use for passing programming verification acknowledgement. For example, if gang programming in the Slave Programming Mode, the slave with SID = 0001 will use Port 3.1 to signal correct or incorrect program verification.
PALE	PROGRAMMING ALE INPUT: Accepted by an 879XBH that is in the Slave Programming Mode. Used to indicate that Ports 3 and 4 contain a command/address.
PROG	PROGRAMMING PULSE: Accepted by 879XBH that is in the Slave Programming Mode. Used to indicate that Ports 3 and 4 contain the data to be programmed. A falling edge on PROG signifies data valid and starts the programming cycle. A rising edge on PROG will halt programming in the slaves.
PACT	PROGRAMMING ACTIVE: Used in the Auto-Programming Mode to indicate when programming activity is complete.
PVER	PROGRAM VERIFIED: A signal output after a programming operation by parts in the Slave Programming Mode.
PDO	PROGRAMMING DURATION OVERFLOWED: A signal output by parts in the Slave Programming Mode. Used to signify that the PROG pulse applied for a programming operation was longer than allowed.
SALE	SLAVE ALE: Output signal from an 879XBH in the Auto Programming Mode. A falling edge on SALE indicates that Ports 3 and 4 contain valid address/command information for slave 879XBHs that may be attached to the master.
SPROG	SLAVE PROGRAMMING PULSE: Output from an 879XBH in the Auto Programming Mode. A falling edge on SPROG indicates that Ports 3 and 4 contain valid data for programming into slave 879XBHs that may be attached to the master.
PORTS 3 and 4	ADDRESS/COMMAND/DATA BUS: Used to pass commands, addresses and data to and from slave mode 879XBHs. Used by chips in the Auto Programming Mode to pass command, addresses and data to slaves. Also used in the Auto Programming Mode as a regular system bus to access external memory. Each line should be pulled up to VCC through a resistor.

Figure 34. Programming Mode Pin Definitions

10.2 Auto Programming Mode

The Auto Programming Mode provides the ability to program the internal 879XBH EPROM without having to use a special EPROM programmer. In this mode, the 879XBH simply programs itself with the data found at external locations 4000H through 5FFFH. All that is required is that some sort of external memory reside at these locations, that EA selects the programming mode and that VPP is applied. Figure 35 shows a minimum configuration for using an 8K x 8 EPROM to program one 879XBH in the Auto Programming Mode.

The 879XBH first reads a word from external memory, then the Modified Quick-Pulse Programming™ Algorithm (described later) is used to program the appropriate EPROM location. Since the erased state of a byte is 0FFH, the Auto Programming Mode will skip locations where the data to be programmed is 0FFH. When all 8K has been programmed, PACT goes high and the part outputs a 0 on Port 3.0 if it programmed correctly and a 1 if it failed.

10.2.1 GANG PROGRAMMING WITH THE AUTO PROGRAMMING MODE

An 879XBH in the Auto Programming Mode can also be used as a programmer for up to 15 other 879XBHs that are configured in the Slave Programming Mode.

To accomplish this, the 879XBH acting as the master outputs the slave command/data pairs on Ports 3 and 4 necessary to program slave parts with the same data it is programming itself with. Slave ALE (SALE) and Slave PROG (SPROG) signals are provided by the master to the slaves to demultiplex the commands from the data. Figure 36 is a block diagram of a gang programming system using one 879XBH in the Auto Programming Mode. The Slave Programming Mode is described in the next section.

The master 879XBH first reads a word from the external memory controlled by ALE, RD and WR. It then drives Ports 3 and 4 with a Data Program command using the appropriate address and alerts the slaves with a falling edge on SALE. Next, the data to be programmed is driven onto Ports 3 and 4 and slave programming begins with a falling edge on SPROG. At the same time, the master begins to program its own EPROM location with the data read in. Intel's Modified Quick-Pulse Programming™ Algorithm is used, with Data Verify commands being given to the slaves after each programming pulse.

When programming is complete PACT goes high and Ports 3 and 4 are driven with all 1s if all parts programmed correctly. Individual bits of Port 3 and 4 will be driven to 0 if the slave with that bit number as an SID did not program correctly. The 879XBH used as the master assigns itself an SID of 0.

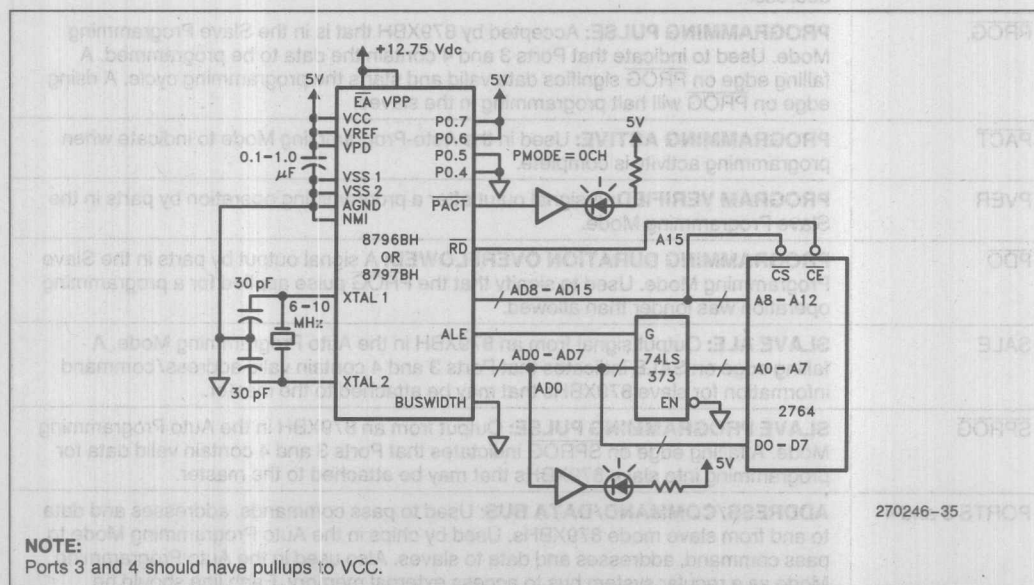


Figure 35. The Auto Programming Mode



17-34

Any number of 879XBHs can be programmed by a master programmer through the Slave Programming Mode.

The programming device uses Ports 3 and 4 of the parts being programmed as a command/data path. The slaves accept signals on PALE (Program ALE) and PROG (Program Enable) to demultiplex the commands and data. The slaves also use PVER, $\overline{\text{PDO}}$ and Ports 3 and 4 to pass error information to the programmer. Support for gang programming of up to 16 879XBHs is provided. If each part is given a unique SID (Slave ID Number) an 879XBH in the Auto Programming Mode can be used as a master to program itself and up to 15 other slave 879XBHs. There is, however, no 879XBH dependent limit to the number of parts that can be gang programmed in the slave mode.

It is important to note that the interface to an 879XBH in the slave mode is similar to a multiplexed bus. Attempting to issue consecutive PALE pulses without a corresponding $\overline{\text{PROG}}$ pulse will produce unexpected results. Similarly, issuing consecutive $\overline{\text{PROG}}$ pulses without the corresponding PALE pulses immediately preceding is equally unpredictable.

10.3.1 SLAVE PROGRAMMING COMMANDS

The commands sent to the slaves are 16-bits wide and contain two fields. Bits 14 and 15 specify the action that the slaves are to perform. Bits 0 through 13 specify the address upon which the action is to take place. Commands are sent via Ports 3 and 4 and are available to cause the slaves to program a word, verify a word, or dump a word (Table 1). The address part of the command sent to the slaves ranges from 2000H to 3FFFH and refers to the internal EPROM memory space. The following sections describe each slave programming mode command.

Mode Commands

P4.7	P4.6	Action
0	0	Word Dump
0	1	Data Verify
1	0	Data Program
1	1	Reserved

DATA PROGRAM COMMAND—After a Data Program Command has been sent to the slaves, $\overline{\text{PROG}}$ must be pulled low to cause the data on Ports 3 and 4 to be programmed into the location specified during the command. The falling edge of $\overline{\text{PROG}}$ is not only used to indicate data valid, but also triggers the hardware programming of the word specified. The slaves will begin programming 48 states after $\overline{\text{PROG}}$ falls, and will continue to program the location until $\overline{\text{PROG}}$ rises.

After the rising edge of $\overline{\text{PROG}}$, the slaves automatically perform a verification of the address just programmed. The result of this verification is then output on PVER (Program Verify) and $\overline{\text{PDO}}$ (Program Duration Overflowed). Therefore, verification information is available following the Data Program Command for programming systems that cannot use the Data Verify command.

If PVER and $\overline{\text{PDO}}$ of all slaves are 1s after $\overline{\text{PROG}}$ rises then the data program was successful everywhere. If PVER is a 0 in any slave, then the data programmed did not verify correctly in that part. If $\overline{\text{PDO}}$ is a 0 in any slave, then the programming pulse in those parts was terminated by an internal safety feature rather than the rising edge of $\overline{\text{PROG}}$. The safety feature prevents over-programming in the slave mode. Figure 37 shows the relationship of PALE, $\overline{\text{PROG}}$, PVER and $\overline{\text{PDO}}$ to the Command/Data Path on Ports 3 and 4 for the Data Program Command.

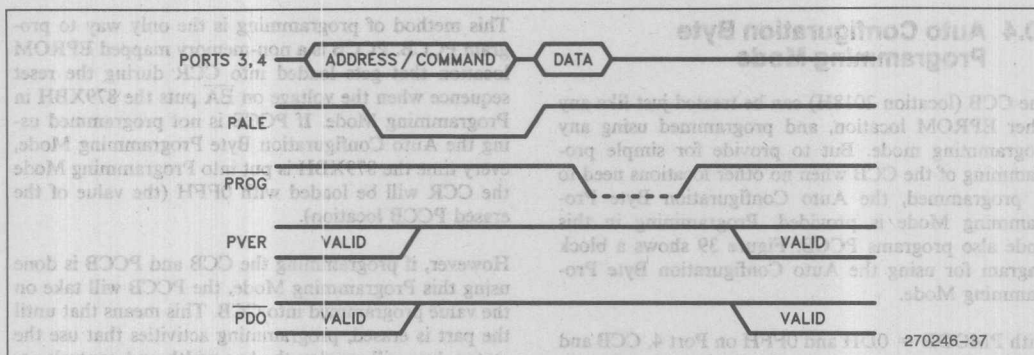


Figure 37. Data Program Signals in Slave Programming Mode

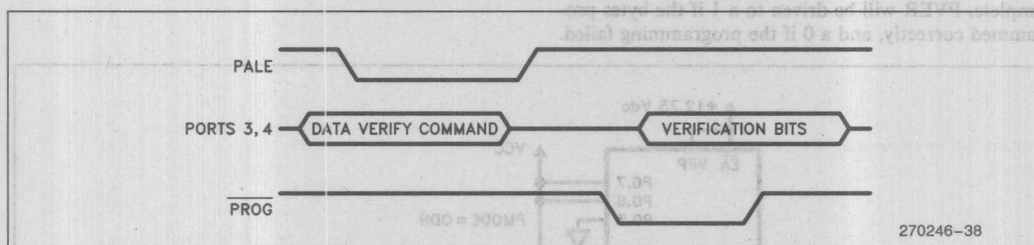


Figure 38. Data Verify Command Signals

DATA VERIFY COMMAND—When the Data Verify Command is sent, the slaves respond by driving one bit of Port 3 and 4 to indicate correct or incorrect verification of the previous Data Program. A 1 indicates correct verification, while a 0 indicates incorrect verification. The SID (Slave ID Number) of each slave determines which bit of the command/data path is driven. PROG from the programmer governs when the slaves drive the bus. Figure 38 shows the relationship of Ports 3 and 4 to PALE and PROG.

This command is always preceded by a Data Program Command in a programming system with as many as 16 slaves. However, a Data Verify Command does not have to follow every Data Program Command.

WORD DUMP COMMAND—When the Word Dump Command is issued, the 879XBH being programmed adds 2000H to the address field of the command and places the value found at the new address on Ports 3 and 4. For example, sending the command #0100H to a slave will result in the slave placing the word found at location 2100H on Ports 3 and 4. PROG from the programmer governs when the slave drives the bus. The signals are the same as shown in Figure 22.

Note that this command will work only when just one slave is attached to the bus, and that there is no restriction on commands that precede or follow a Word Dump Command.

10.3.2 GANG PROGRAMMING WITH THE SLAVE PROGRAMMING MODE

Gang programming of 879XBHs can be done using the Slave Programming Mode. There is no 879XBH based limit on the number of chips that may be hooked to the same Port 3/Port 4 data path for gang programming.

If more than 16 chips are being gang programmed, the PVER and PDO outputs of each chip could be used for verification. The master programmer could issue a data program command then either watch every chip's error signals, or AND all the signals together to get a system PVER and PDO.

If 16 or fewer 879XBHs are to be gang programmed at once, a more flexible form of verification is available. By giving each chip being programmed a unique SID, the master programmer could then issue a data verify command after the data program command. When a verify command is seen by the slaves, each will drive one pin of Port 3 or 4 with a 1 if the programming verified correctly or a 0 if programming failed. The SID is used by each slave to determine which Port 3, 4 bit it is assigned. An 879XBH in the Auto Programming Mode could be the master programmer if 15 or fewer slaves need to be programmed (see Gang Programming with the Auto Programming Mode).

10.4 Auto Configuration Byte Programming Mode

The CCB (location 2018H) can be treated just like any other EPROM location, and programmed using any programming mode. But to provide for simple programming of the CCB when no other locations need to be programmed, the Auto Configuration Byte Programming Mode is provided. Programming in this mode also programs PCCB. Figure 39 shows a block diagram for using the Auto Configuration Byte Programming Mode.

With PMODE = 0DH and 0FFH on Port 4, CCB and PCCB will be programmed to the value on Port 3 when a logic 0 is placed on PALE. After programming is complete, PVER will be driven to a 1 if the bytes programmed correctly, and a 0 if the programming failed.

This method of programming is the only way to program PCCB. PCCB is a non-memory mapped EPROM location that gets loaded into CCR during the reset sequence when the voltage on EA puts the 879XBH in Programming Mode. If PCCB is not programmed using the Auto Configuration Byte Programming Mode, every time the 879XBH is put into Programming Mode the CCR will be loaded with 0FFH (the value of the erased PCCB location).

However, if programming the CCB and PCCB is done using this Programming Mode, the PCCB will take on the value programmed into CCB. This means that until the part is erased, programming activities that use the system bus will employ the bus width and controls selected by the user's CCB.

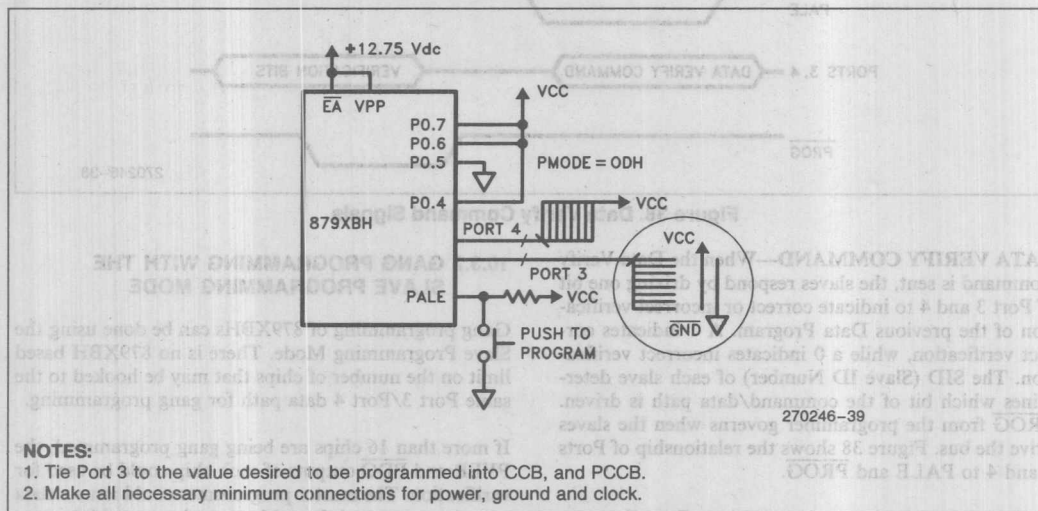


Figure 39. The Auto CCR Programming Mode

10.5 Run-Time Programming

Run-Time Programming of the 879XBH is provided to allow the user complete flexibility in the ways in which the internal EPROM is programmed. That flexibility includes the ability to program just one byte or one word instead of the whole EPROM, and extends to the hardware necessary to program. The only additional requirement of a system is that a programming voltage is applied to VPP. Run-Time Programming is done with EA at TTL-high (normal operation—internal/external access).

To Run-Time program, the user writes a byte or word to the location to be programmed. Once this is done, the 879XBH will continue to program that location until another data read from or data write to the EPROM occurs. The user can therefore control the duration of the programming pulse to within a few microseconds. An intelligent algorithm should be implemented in software. It is recommended that the Modified Quick-Pulse Programming Algorithm be implemented.

After the programming of a location has started, care must be taken to ensure that no program fetches (or

pre-fetches) occur from internal memory. This is of no concern if the program is executing from external memory. However, if the program is executing from internal memory when the write occurs, it will be necessary to use the built in "Jump to Self" located at 201AH.

"Jump to Self" is a two byte instruction in the Intel test ROM which can be CALLED after the user has started programming a location by writing to it. A software timer interrupt could then be used to escape from the "Jump to Self" when the proper programming pulse duration has elapsed. Figure 40 is an example of how to program an EPROM location while execution is entirely internal.

Upon entering the PROGRAM routine, the address and data are retrieved from the STACK and a Software Timer is set to expire one programming pulse later. The data is then written to the EPROM location and a CALL to location 201AH is made. Location 201AH is in Intel reserved test ROM, and contains the two byte opcode for a "Jump to Self". The minimum interrupt service routine would remove the 201AH return address from the STACK and return.

PROGRAM:

```
POP temp
POP address_temp
POP data_temp
PUSH temp

PUSHF
LDB int_mask , #enable_swt_only
LDB HSO_COMMAND , #SWT0_ovf
ADD HSO_TIME,TIMER1, #program_pulse

EI
ST data-temp, [address_temp]
CALL 201AH

POPF
RET

SWT_ISR:
...

swt0_expired:
POP 0
RET
...
```

```
;take parameters from the
STACK

;save current status
;enable only swt interrupts
;load swt command to interrupt
;when program pulse time
;has elapsed
```

Figure 40. Programming the EPROM from Internal Memory Execution

10.6 ROM/EPROM Program Lock

Protection mechanisms have been provided on the ROM and EPROM versions of the 8096BH to inhibit unauthorized accesses of internal program memory. However, there must always be a way to allow authorized program memory dumps for testing purposes. The following describes 839XBH, 879XBH program lock features and the mode provided for authorized memory dumps.

10.6.1 LOCK FEATURES

Write protection is provided for EPROM parts, while READ protection is provided for both ROM and EPROM parts.

Write protection is enabled by causing the LOCO bit in the CCR to take the value 0. When WRITE protection is selected, the bus controller will cycle through the write sequence, but will not actually drive data to the EPROM and will not enable VPP to the EPROM. This protects the entire EPROM 2000H–3FFFH from inadvertent or unauthorized programming, and also prevents writes to the EPROM from upsetting program execution. If write protection is not enabled, a data write to an internal EPROM location will begin programming that location, and continue programming the location until a data read of the internal EPROM is executed. While programming, instruction fetches from internal EPROM will not be successful.

READ protection is selected by causing the LOC1 bit in the CCR to take the value 0. When READ protection is enabled, the bus controller will only perform a data read from the address range 2020H–3FFFH if the slave program counter is in the range 2000H–3FFFH. Note that since the slave PC can be many bytes ahead of the CPU program counter, an instruction that is located after address 3FFAH may not be allowed to access protected memory, even though the instruction is itself protected.

If the bus controller receives a request to perform a READ of protected memory, the READ sequence occurs with indeterminant data being returned to the CPU.

Other enhancements were also made to the 8096BH for program protection. For example, the value of EA is latched on reset so that the device cannot be switched from external to internal execution mode at run-time. In addition, if READ protection is selected, an NMI event will cause the device to switch to external only execution mode. Internal execution can only resume by resetting the chip.

10.6.2 AUTHORIZED ACCESS OF PROTECTED MEMORY

To provide a method of dumping the internal ROM/EPROM for testing purposes a "Security Key" mechanism and ROM dump mode have been implemented.

The security key is a 128 bit number, located in internal memory, that must be matched before a ROM dump will occur. The application code contains the security key starting at location 2020H.

The ROM dump mode is entered just like any programming mode ($\overline{\text{EA}} = 12.75\text{V}$), except that a special PMODE strapping is used. The PMODE for ROM dump is 6H (0110B).

The ROM dump sequence begins with a security key verification. Users must place at external locations 4020H–402FH the same 16 byte key that resides inside the chip at locations 2020H–202FH. Before doing a ROM dump, the chip checks that the keys match.

After a successful key verification, the chip dumps data to external locations 1000H–11FFH and 4000H–5FFFH. Unspecified data appears at the low addresses.

Internal EPROM/ROM is dumped to 4000H–5FFFH, beginning with internal address 2000H.

If a security key verification is not successful, the chip will put itself into an endless loop of internal execution.

NOTE:

Substantial effort has been expended to provide an excellent program protection scheme. However, Intel cannot, and does not guarantee that the protection methods that we have devised will prevent unauthorized access.

10.7 Modified Quick-Pulse Programming™ Algorithm

The Modified Quick-Pulse Programming Algorithm calls for each EPROM location to receive 25 separate $100\ \mu\text{s}$ ($\pm 5\ \mu\text{s}$) program cycles. Verification of correct programming is done after the 25 pulses. If the location verifies correctly, the next location is programmed. If the location fails to verify, the location has failed.

Once all locations are programmed and verified, the entire EPROM is again verified.

Programming of 879XBH parts is done with $V_{PP} = 12.75\text{V} \pm 0.25\text{V}$ and $V_{CC} = 5.0\text{V} \pm 0.5\text{V}$.

10.8 Signature Word

The 8X9XBH contains a signature word at location 2070H. The word can be accessed in the slave mode by executing a word dump command.

Table 2. 8X9XBH Signature Words

Device	Signature Word
879XBH	896FH
839XBH	896EH
809XBH	Undefined

10.9 Erasing the 879XBH EPROM

Initially, and after each erasure, all bits of the 879XBH are in the "1" state. Data is introduced by selectively

programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The erasure characteristics of the 879XBH are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms (\AA). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000 \AA range. Constant exposure to room level fluorescent lighting could erase the typical 879XBH in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 879XBH is to be exposed to light for extended periods of time, opaque labels must be placed over the EPROM's window to prevent unintentional erasure.

The recommended erasure procedure for the 879XBH is exposure to shortwave ultraviolet light which has a wavelength of 2537 \AA . The integrated dose (i.e., UV intensity \times exposure time) for erasure should be a minimum of 15 Wsec/cm². The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 $\mu\text{W}/\text{cm}^2$ power rating. The 879XBH should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated dose an 879XBH can be exposed to without damage is 7258 Wsec/cm² (1 week @ 12000 $\mu\text{W}/\text{cm}^2$). Exposure of the 879XBH to high intensity UV light for long periods may cause permanent damage.

80C196KA Architectural Overview

18

ADVANCED CMOS MICROCONTROLLER ARCHITECTURAL OVERVIEW

AUTOMOTIVE

1.0 INTRODUCTION

All of the features available on the standard MCS®-96 are present on the 80C196KA including:

- Register to Register Architecture
- 232 Bytes of Register File
- 22 Interrupt Sources With 8 Vector Locations
- High Speed 16x16 Multiply
- High Speed 32/16 Divide
- Five 8-bit I/O Ports
- Analog to Digital Converter (A/D Versions Only)
- Pulse-Width-Modulated Output
- Full Duplex Serial Port With Dedicated Baud Rate Generator
- 16-bit Watchdog Timer
- High Speed Subsystem With
 - Up to 4 Time Capture Inputs
 - Up to 6 Time Triggered Outputs
- 2 16-bit Timer/Counters
- 4 Software Timers

In addition, the 80C196KA has:

- Independent Capture of Timer2
- Up and Down Counting on Timer2
- 2.8 μ s 16x16 Multiply vs 6.25 μ s on 8096BH

4.8 μ s 32/16 Divide vs 6.25 μ s on 8096BH

6 Additional Interrupt Sources / 10 Additional Vectors

6 Additional Instructions

Power Down and Idle Modes for Power Savings

and many other feature enhancements. The 80C196KA can be plugged into most 8096BH designs with only a few minor software changes.

This document can be used as a stand-alone guide to the features of the 80C196KA and as a programmer's guide and user's manual by experienced 8096 programmers. For those people who are not familiar with the details of programming an 8096, this manual should be used in conjunction with the current edition of the Embedded Controller Handbook.

2.0 ARCHITECTURAL OVERVIEW

For the purpose of describing its operation, the 80C196KA can be divided into three sections: the processing unit, peripheral (I/O) devices, and support circuitry. The processing unit consists of the 16-bit CPU with its register file, the interrupt controller and the memory controller. Peripheral devices, a clock generator, and some miscellaneous support circuitry make up the remainder of the chip. A block diagram of the 80C196KA is shown in Figure 1.

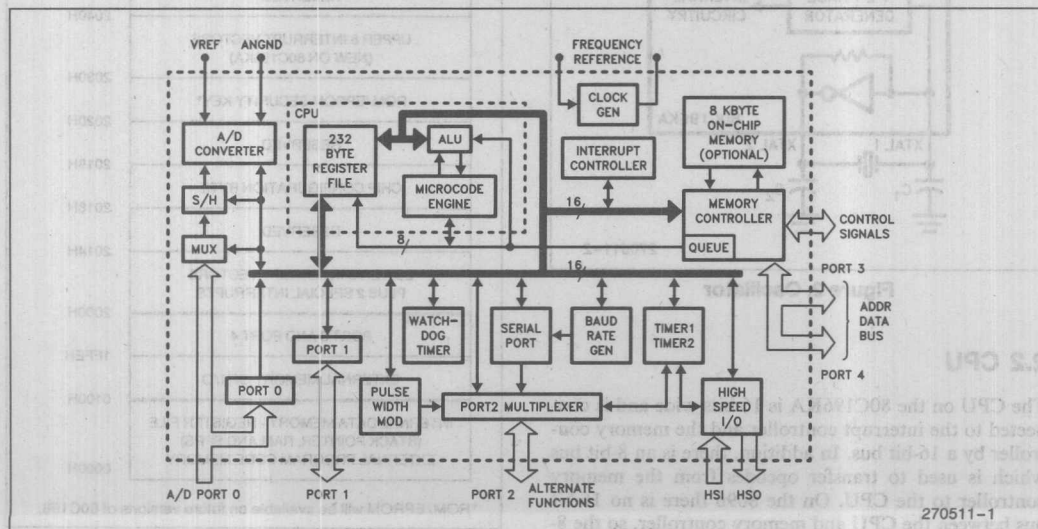


Figure 1. 80C196KA Block Diagram

2.1 INTERNAL TIMINGS

Internal operation of the chip is based on the oscillator frequency divided by two, giving the basic operating time unit, known as a "State Time". With a 10 MHz oscillator, a state time is 200 ns. With an 8 MHz oscillator, a state time is 250 ns, the same as that of an 8096BH running with a 12 MHz oscillator. Since the 80C196KA will be run at many frequencies, the times given throughout this overview will be in state times or "states", unless otherwise specified.

Either a crystal or an external source can be used to drive the on-chip oscillator. Figure 2 shows a circuit for the oscillator connected to a crystal. When an external source is used, it is connected to the XTAL1 pin leaving the XTAL2 pin floating. The XTAL2 pin becomes a weak output in this mode and must be left unconnected.

Two non-overlapping internal phases are created by the clock generator: phase 1 and phase 2. Phase 2 is buffered and output on the CLKOUT pin. This is not the same as on the 8096BH, since it uses a three-phase clock. Changing from a three-phase clock to a two-phase one speeds up the operation of the chip for a set oscillator frequency. It should cause no compatibility problems in most designs, but does cause some differences in the system bus timings. A detailed description of the bus timing is included in the electrical characteristics section of this document.

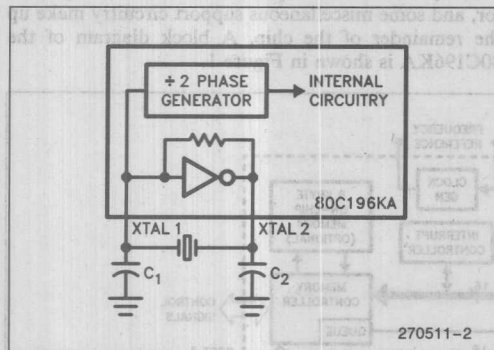


Figure 2. Oscillator

2.2 CPU

The CPU on the 80C196KA is 16 bits wide and is connected to the interrupt controller and the memory controller by a 16-bit bus. In addition, there is an 8-bit bus which is used to transfer opcodes from the memory controller to the CPU. On the 8096 there is no 16-bit bus between the CPU and memory controller, so the 8-bit bus is used for both data and opcode transfers. All of the peripheral devices on the 80C196KA are connected to the CPU by a 16-bit bus.

A microcode engine controls the CPU, allowing it to perform operations with any byte, word or double word in the 232-byte Register File. Operations can also be performed with any of the I/O Control Registers, also called Special Function Registers (SFRs). With a flat architecture, the programmer is not limited to a single accumulator since all 256 bytes in the register file and SFR space can be used as accumulators. This eliminates accumulator bottleneck and allows the use of 3 operand instructions. The internal hardware of the CPU is similar to that of the 8096, except that extra hardware has been added to provide a faster multiply.

2.3 MEMORY MAP

64 Kbytes of addressable memory space are available on the 80C196KA, most of which can be used for program or data storage. The space from 100H through 0FFFFH contains a small block of reserved or special function locations but is otherwise available to the user. The reserved locations must contain 0FFH. Resetting the chip sets the program counter to location 2080H, allowing 8 Kbytes of RAM contiguous with the internal RAM at location 0FFH. The interrupt vectors, configuration byte, and several reserved addresses are located between 2000H and 207FH. Figure 3 shows a memory map of the 80C196KA memory space.

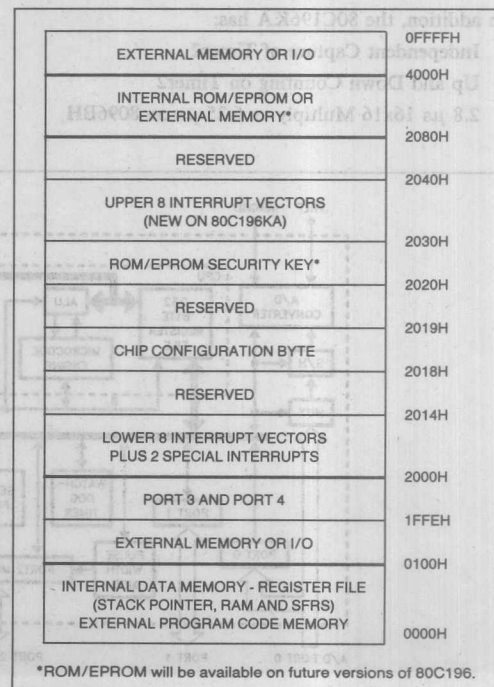


Figure 3. 80C196KA Memory Map

Between 0H and 0FFH program execution fetches will always be from external memory, even if the chip has an onboard ROM or EPROM. This area of external memory is reserved for use by Intel development systems and should not be used in applications which will require development tools. Data fetches will always come from the on-chip register file and SFRs. The internal RAM from location 01AH (26 decimal) to 0FFH is the register file. This memory region, as well as the status of the majority of the chip, is kept alive while the chip is in the powerdown mode. (On the 8096 only the top 16 bytes of RAM were kept alive.) Details on powerdown mode are discussed in a later section.

Locations 18H and 19H are considered part of the register file although they are used as the stack pointer. The stack can be located anywhere in memory, internal or external, by using the 16-bit pointer. If the stack is not being used, these two bytes can be used as regular RAM.

Locations 00H through 17H are the I/O control registers or SFRs. As shown in Figure 4, two SFR windows are provided on the 80C196KA. Selecting the active window is done by using the Window Select Register (WSR) at location 14H in all of the windows.

Only two values may be written to the WSR, 0 and 15. Other values are reserved for use in future parts and will cause unpredictable operation.

Window 0, the register window selected with WSR=0, is a superset of the one used on the 8096. As depicted in Figure 5, it has 24 registers, some of which have different functions when read than when written. Figure 6 contains brief descriptions of the registers. Detailed descriptions are contained in the section which discusses the peripheral device controlled by the register.

In register Window 15 (WSR=15), the operation of the SFRs is changed, so that those which were read-only in the 8096 SFR space are write-only and vice versa. The only exception to this is that TIMER2 is read/write in Window 0, and T2 Capture is read/write in Window 15. Registers which can be read and written in Window 0 can also be read and written in Window 15. Details of using Window 15 are discussed in the peripheral description section.

Caution must be taken when using the SFRs as sources of operations or as base or index registers for indirect or indexed operations. It is possible to not get the desired results, since external events can change SFRs and some SFRs clear when read. The potential for an SFR to change value must be taken into account when operating on these registers. This is particularly important when high level languages are used as they do not always make allowances for SFR-type registers.

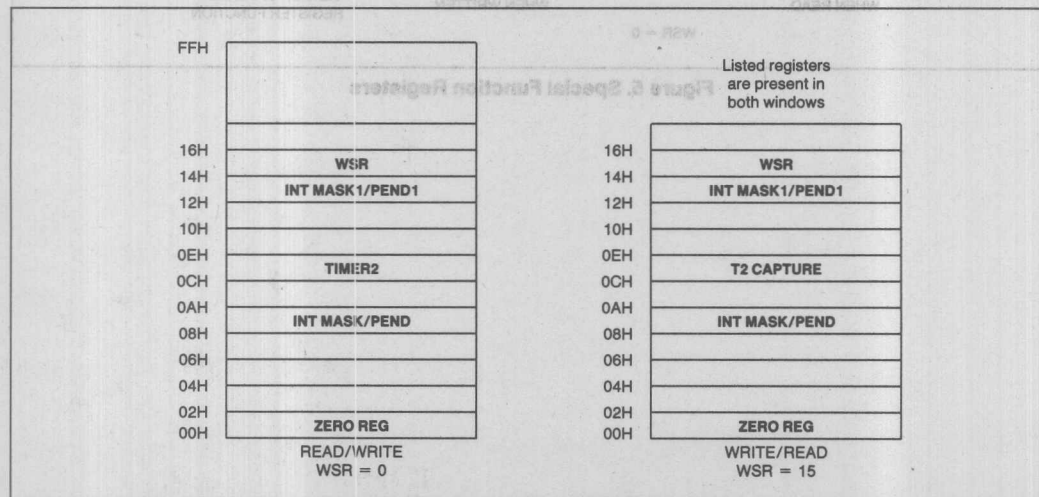


Figure 4. Multiple Register Windows

19H	STACK POINTER	19H	STACK POINTER	
18H		18H		
17H	*IOS2	17H	PWM_CONTROL	
16H	IOS1	16H	IOC1	
15H	IOS0	15H	IOC0	
14H	*WSR	14H	*WSR	
13H	*INT_MASK 1	13H	*INT_MASK 1	
12H	*INT_PEND 1	12H	*INT_PEND 1	
11H	*SP_STAT	11H	*SP_CON	
10H	PORT2	10H	PORT2	
0FH	PORT1	0FH	PORT1	
0EH	PORT0	0EH	BAUD RATE	
0DH	TIMER2 (HI)	0DH	TIMER2 (HI)	0DH *T2 CAPTURE (HI)
0CH	TIMER2 (LO)	0CH	TIMER2 (LO)	0CH *T2 CAPTURE (LO)
0BH	TIMER1 (HI)	0BH	*IOC2	WSR = 15
0AH	TIMER1 (LO)	0AH	WATCHDOG	
09H	INT_PENDING	09H	INT_PENDING	OTHER SFRS IN WSR 15 BECOME READABLE IF THEY WERE WRITABLE IN WSR = 0 AND WRITABLE IF THEY WERE READABLE IN WSR = 0
08H	INT_MASK	08H	INT_MASK	
07H	SBUF(RX)	07H	SBUF(TX)	
06H	HSI_STATUS	06H	HSO_COMMAND	
05H	HSI_TIME (HI)	05H	HSO_TIME (HI)	
04H	HSI_TIME (LO)	04H	HSO_TIME (LO)	
03H	AD_RESULT (HI)	03H	HSI_MODE	
02H	AD_RESULT (LO)	02H	AD_COMMAND	
01H	ZERO REG (HI)	01H	ZERO REG (HI)	
00H	ZERO REG (LO)	00H	ZERO REG (LO)	

WHEN READ

WHEN WRITTEN

*NEW OR CHANGED REGISTER FUNCTION

WSR = 0

Figure 5. Special Function Registers

19H	WSR	19H	WSR
18H	INT_MASK_PEND1	18H	INT_MASK_PEND1
17H		17H	
16H		16H	
15H		15H	
14H		14H	
13H		13H	
12H		12H	
11H		11H	
10H		10H	
09H		09H	
08H		08H	
07H		07H	
06H		06H	
05H		05H	
04H		04H	
03H		03H	
02H		02H	
01H		01H	
00H		00H	
READWRITE WSR = 0		READWRITE WSR = 0	

Figure 4. Multiple Register Windows

Register	Description
R0	Zero Register - Always reads as a zero, useful for a base when indexing and as a constant for calculations and compares.
AD_RESULT	A/D Result Hi/Low - Low and high order results of the A/D converter
AD_COMMAND	A/D Command Register - Controls the A/D
HSI_MODE	HSI Mode Register - Sets the mode of the High Speed Input unit.
HSI_TIME	HSI Time Hi/Lo - Contains the time at which the High Speed Input unit was triggered.
HSO_TIME	HSO Time Hi/Lo - Sets the time or count for the High Speed Output to execute the command in the Command Register.
HSO_COMMAND	HSO Command Register - Determines what will happen at the time loaded into the HSO Time registers.
HSI_STATUS	HSI Status Registers - Indicates which HSI pins were detected at the time in the HSI Time registers and the current state of the pins.
SBUF(TX)	Transmit buffer for the serial port, holds contents to be outputted.
SBUF(RX)	Receive buffer for the serial port, holds the byte just received by the serial port.
INT_MASK	Interrupt Mask Register - Enables or disables the individual interrupts (also IMASK).
INT_PENDING	Interrupt Pending Register - Indicates that an interrupt signal has occurred on one of the sources and has not been serviced (also IPEND).
WATCHDOG	Watchdog Timer Register - Written to periodically to hold off automatic reset every 64K state times.
TIMER1	Timer 1 Hi/Lo - Timer1 high and low bytes.
TIMER2	Timer 2 Hi/Lo - Timer2 high and low bytes.
IOPORT0	Port 0 Register - Levels on pins of Port 0.
BAUD_RATE	Register which determines the baud rate, this register is loaded sequentially.
IOPORT1	Port 1 Register - Used to read or write to Port 1.
IOPORT2	Port 2 Register - Used to read or write to Port 2.
SP_STAT	Serial Port Status - Indicates the status of the serial port.
SP_CON	Serial Port Control - Used to set the mode of the serial port.
IOS0	I/O Status Register 0 - Contains information on the HSO status.
IOS1	I/O Status Register 1 - Contains information on the status of the timers and of the HSI.
IOC0	I/O Control Register 0 - Controls alternate functions of HSI pins, Timer 2 reset sources and Timer 2 clock sources.
IOC1	I/O Control Register 1 - Controls alternate functions of Port 2 pins, timer interrupts and HSI interrupts.
PWM_CONTROL	Pulse Width Modulation Control Register - Sets the duration of the PWM pulse.
IPEND1	Interrupt Pending register for the 8 new interrupt vectors (also INT_PENDING1)
IMASK1	Interrupt Mask register for the 8 new interrupt vectors (also INT_MASK1)
IOC2	I/O Control Register 2 - Controls new 80C196KA features
IOS2	I/O Status Register 2 - Contains information on HSO events
WSR	Window Select Register - Selects register window

Figure 6. Special Function Register Description

2.4 MEMORY CONTROLLER

All of the program memory and the external data memory are transferred to the CPU through the memory controller. Within the memory controller is a slave program counter, an instruction queue, and a bus controller.

The slave program counter keeps track of the program counter in the CPU and requests the correct sequence of instructions to be fetched by the bus controller and stored in the queue.

Instruction Queue

A four byte instruction queue allows the CPU to run faster by keeping the next instruction byte almost always available. When the instruction flow changes, as with a branch or call instruction, the queue is flushed and refilled. The amount of time required to do this is included in the instruction execution times which are listed in other sections of this document.

When debugging code using a logic analyzer, one must be aware of the queue. It is not possible to determine when an instruction will begin executing by simply watching when it is read since the queue is filled in advance of instruction execution. In addition, the algorithms which are used to keep the queue full may cause instructions to be read into the 80C196KA multiple times.

Bus Controller

Both 8-bit and 16-bit bus modes are supported by the bus controller. A block diagram of the two modes is shown in Figure 7. Each mode has several variations, all of which are controlled by the Chip Configuration Register (CCR), shown in Figure 8. This register is at an unmapped location within the 80C196KA and is loaded from location 2018H during the chip reset sequence.

Switching between 8 and 16-bit bus modes can be done using the buswidth pin if the CCR is set for a 16-bit bus. Dynamically switching between the two modes is possible by changing this pin on the fly. A system using 16-bit wide program memory for speed, but only needing one 8-bit RAM chip, could make use of this feature to avoid the use of another RAM or the software needed to convert word wide data into data stored in every other byte.

When CCR bits 2 and 3 are both set to 1 the standard MCS-96 bus control signals are provided, as shown in Figure 9. \overline{WR} will come out for each write. \overline{BHE} will be valid throughout the bus cycle and can be combined with the \overline{WR} and address line 0 to form \overline{WRL} (Write Low byte) and \overline{WRH} (Write High byte). \overline{ALE} will rise as the address starts to come out and will fall to provide a signal to externally latch the address.

The Write Strobe mode eliminates the need to externally decode \overline{WRL} and \overline{WRH} (See Figure 10). In 16-bit bus modes, \overline{WRL} and \overline{WRH} are provided on the \overline{WR}

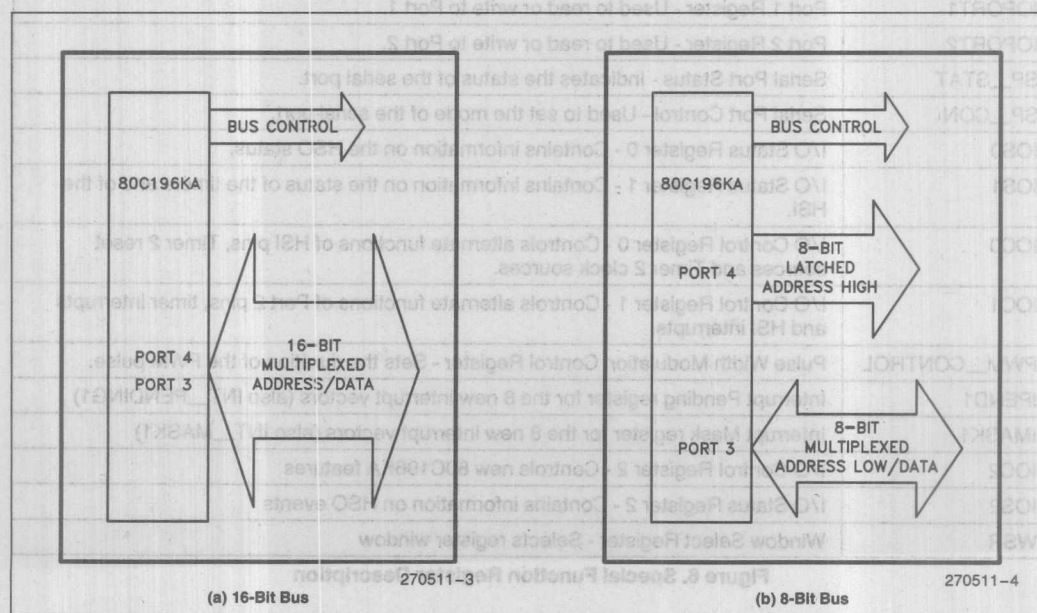


Figure 7. Bus Width Options

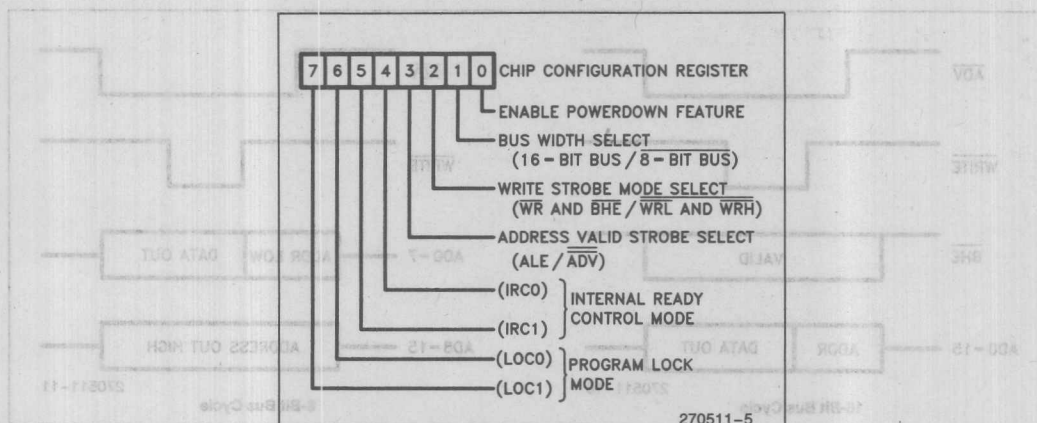


Figure 8. Format of the Chip Configuration Register

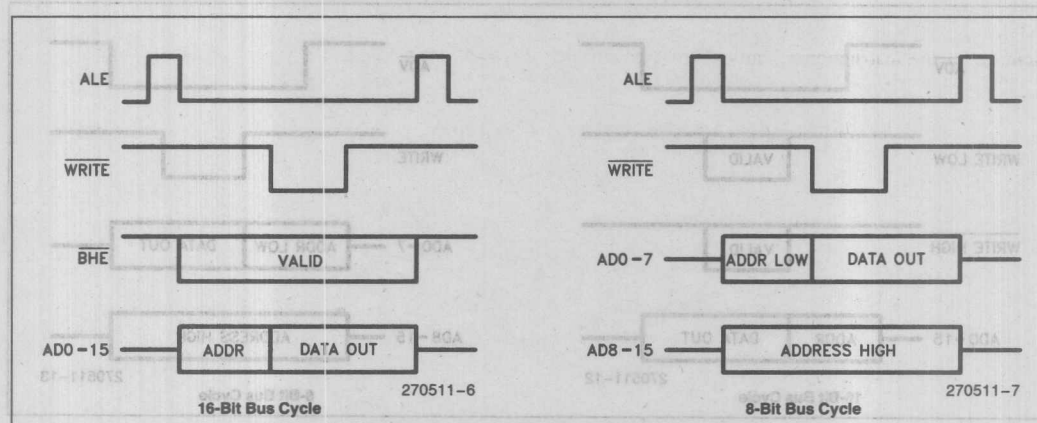


Figure 9. Standard Bus Control

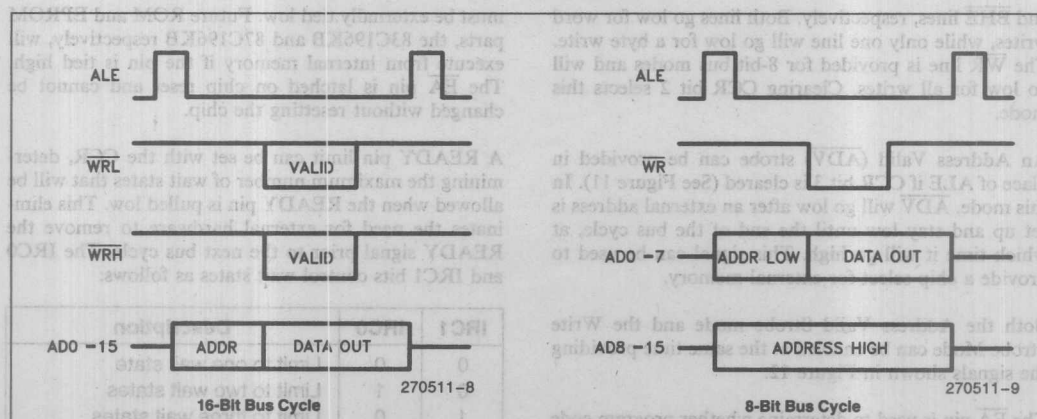


Figure 10. Write Strobe Mode

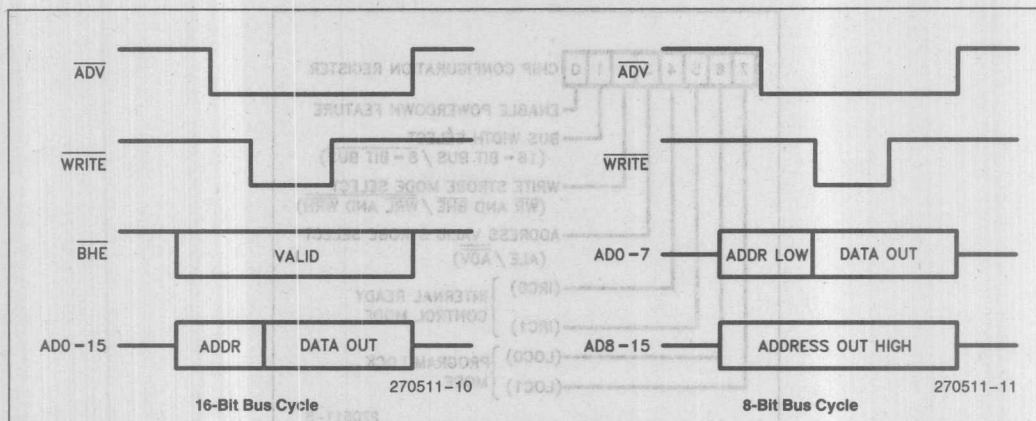


Figure 11. Address Valid Mode

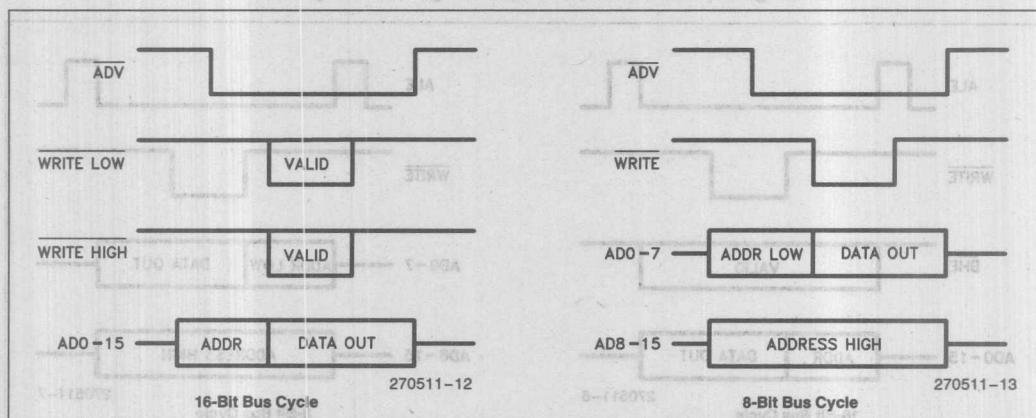


Figure 12. Address Valid With Write Strobe Mode

and $\overline{\text{BHE}}$ lines, respectively. Both lines go low for word writes, while only one line will go low for a byte write. The $\overline{\text{WR}}$ line is provided for 8-bit bus modes and will go low for all writes. Clearing CCR bit 2 selects this mode.

An Address Valid ($\overline{\text{ADV}}$) strobe can be provided in place of $\overline{\text{ALE}}$ if CCR bit 3 is cleared (See Figure 11). In this mode, $\overline{\text{ADV}}$ will go low after an external address is set up and stay low until the end of the bus cycle, at which time it will go high. This signal can be used to provide a chip select for external memory.

Both the Address Valid Strobe mode and the Write Strobe Mode can be enabled at the same time providing the signals shown in Figure 12.

The $\overline{\text{EA}}$ pin is used to determine whether program code in the address range 2000H through 3FFFH is fetched from internal memory or external memory. Since the 80C196KA does not have internal memory this pin

must be externally tied low. Future ROM and EPROM parts, the 83C196KB and 87C196KB respectively, will execute from internal memory if the pin is tied high. The $\overline{\text{EA}}$ pin is latched on chip reset and cannot be changed without resetting the chip.

A $\overline{\text{READY}}$ pin limit can be set with the CCR, determining the maximum number of wait states that will be allowed when the $\overline{\text{READY}}$ pin is pulled low. This eliminates the need for external hardware to remove the $\overline{\text{READY}}$ signal prior to the next bus cycle. The IRC0 and IRC1 bits control wait states as follows:

IRC1	IRC0	Description
0	0	Limit to one wait state
0	1	Limit to two wait states
1	0	Limit to three wait states
1	1	Wait states not limited internally

When internal program memory is used, the CCR can set read and write protection using the LOC0 and LOC1 bits (CCR bits 6 and 7). A zero on LOC0 enables read protections and a zero on LOC1 enables write protection. Both read and write protection may be enabled at the same time by clearing both bits.

2.5 INTERRUPTS

Twenty-eight (28) sources of interrupts are available on the 80C196KA. These sources are gathered into 15 vectors plus special vectors for NMI, the TRAP instruction, and Unimplemented Opcodes. Figure 13 shows the routing of the interrupt sources into their vectors as well as the control bits which enable some of the sources.

NMI, the external Non-Maskable Interrupt, is the highest priority peripheral interrupt. It vectors indirectly through location 203EH. For design symmetry, a mask bit exists in INT_MASK1 for the NMI. To prevent accidental masking of an NMI, the bit does not function and will not stop an NMI from occurring.

Opcode F7H, the TRAP instruction, causes an indirect vector through location 2010H. All unimplemented opcodes are mapped into a special interrupt vector through location 2012H. They act as uninterruptable instructions and take one more state time than the TRAP instruction.

The interrupt sources in the 80C196KA are arranged in a fixed priority. Figure 14 shows the priorities (15 is highest) of the interrupts and their vector locations. If simultaneous interrupt requests are received, the highest priority source that is both pending and enabled will get serviced. Software priorities can be provided by enabling and disabling different interrupts in different routines. When an interrupt occurs, the 80C196KA's response is identical to that of the 8096; it decrements the stack pointer value by 2 and then stacks the program counter value. Because of the additional 16-bit internal bus, the 80C196KA interrupt response takes only 16/18 states (states: stack internal/external).

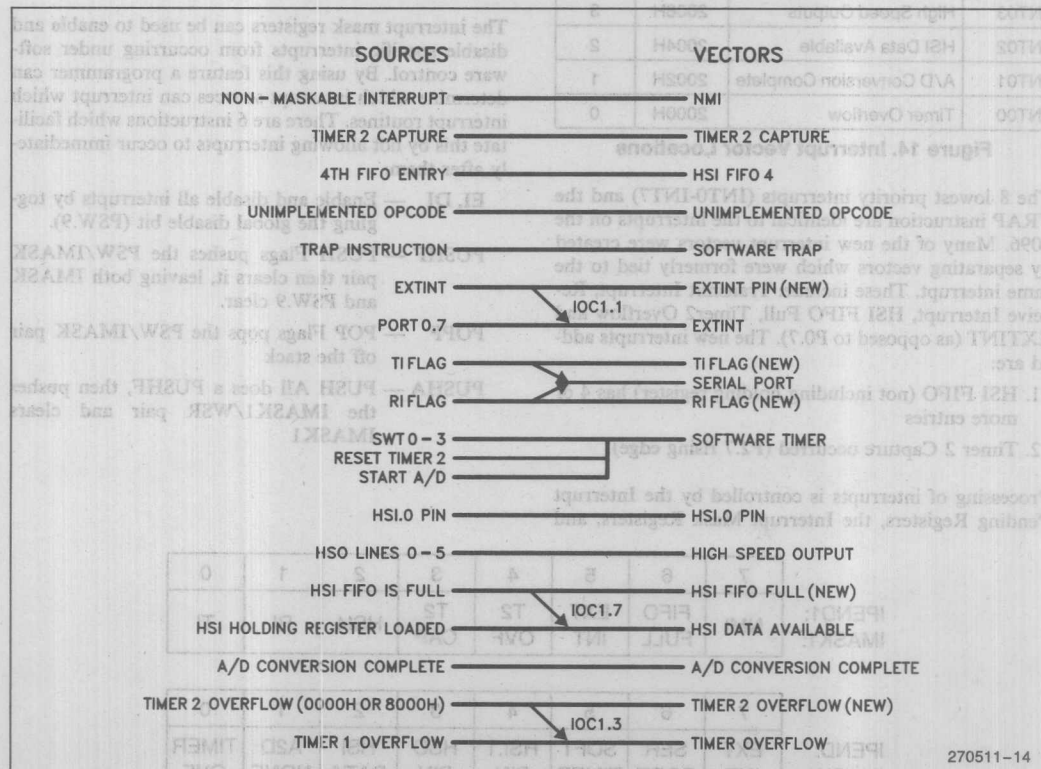


Figure 13. All Possible Interrupt Sources

80C196KA INTERRUPTS

Number	Source	Vector Location	Priority
INT15	NMI	203EH	15
INT14	HSI FIFO Full	203CH	14
INT13	EXTINT Pin	203AH	13
INT12	TIMER2 Overflow	2038H	12
INT11	TIMER2 Capture	2036H	11
INT10	4th Entry into HSI FIFO	2034H	10
INT09	RI	2032H	9
INT08	TI	2030H	8
SPECIAL	Unimplemented Opcode	2012H	N/A
SPECIAL	Trap	2010H	N/A
INT07	EXTINT	200EH	7
INT06	Serial Port	200CH	6
INT05	Software Timer	200AH	5
INT04	HSI.0 Pin	2008H	4
INT03	High Speed Outputs	2006H	3
INT02	HSI Data Available	2004H	2
INT01	A/D Conversion Complete	2002H	1
INT00	Timer Overflow	2000H	0

Figure 14. Interrupt Vector Locations

The 8 lowest priority interrupts (INT0-INT7) and the TRAP instruction are identical to the interrupts on the 8096. Many of the new interrupt vectors were created by separating vectors which were formerly tied to the same interrupt. These include: Transmit Interrupt, Receive Interrupt, HSI FIFO Full, Timer2 Overflow and EXTINT (as opposed to P0.7). The new interrupts added are:

1. HSI FIFO (not including holding register) has 4 or more entries
2. Timer 2 Capture occurred (P2.7 rising edge).

Processing of interrupts is controlled by the Interrupt Pending Registers, the Interrupt Mask Registers, and

the Global Disable Bit. The Interrupt Pending Registers (shown in Figure 15) have one bit for each interrupt vector. If a transition occurs to trigger a particular interrupt, the associated bit in the pending register is set. When a vector to an interrupt routine is taken, the associated pending bit is cleared.

The Interrupt Mask Registers (IMASK, IMASK1) have bits to correspond to each interrupt and are set up identically to the Interrupt Pending Registers. Each mask bit can be set or cleared in software to enable or disable individual interrupts. These registers are also referred to as INT_MASK and INT_MASK1.

PSW bit 9, the global Interrupt Disable Bit, controls the entire interrupt structure. When it is cleared, all interrupts are disabled except NMI, TRAP and unimplemented opcode. When it is set and an interrupt is both pending and unmasked (Ipend.x = 1, Imask.x = 1), the interrupt service procedure begins. The highest priority interrupt which is pending and unmasked is the first to occur. Interrupt servicing involves a call to the address stored in the interrupt vector location and clearing of the interrupt pending bit.

The interrupt mask registers can be used to enable and disable specific interrupts from occurring under software control. By using this feature a programmer can determine which interrupt sources can interrupt which interrupt routines. There are 6 instructions which facilitate this by not allowing interrupts to occur immediately after them:

- EI, DI — Enable and disable all interrupts by toggling the global disable bit (PSW.9).
- PUSHF — PUSH Flags pushes the PSW/IMASK pair then clears it, leaving both IMASK and PSW.9 clear.
- POPF — POP Flags pops the PSW/IMASK pair off the stack
- PUSHA — PUSH All does a PUSHF, then pushes the IMASK1/WSR pair and clears IMASK1

	7	6	5	4	3	2	1	0
IPEND1:	NMI	FIFO	EXT	T2	T2	HSI4	RI	TI
IMASK1:		FULL	INT	OVF	CAP			

	7	6	5	4	3	2	1	0
IPEND:	EXT	SER	SOFT	HSI.1	HSO	HSI	A2D	TIMER
IMASK1:	INT	PORT	TIMER	PIN	PIN	DATA	NONE	OVF

Figure 15. Interrupt Pending Registers

POPA — POP All pops the IMASK1/WSR pair and then does a POPF

Interrupts can also not occur immediately after execution of any unimplemented opcode.

TRAP — The software trap instruction

SIGND — The signed prefix for multiply and divide instructions

PUSHA, PUSHF, and DI disable interrupts until software changes either the interrupt mask, PSW.9 or both. POPA, POPF, and EI can enable interrupts and are frequently used at the end of an interrupt routine, just prior to a RETurn. By preventing interrupts from occurring between these instructions and a RETurn, the RET is always executed and the stack will not build up needlessly.

Interrupts cannot occur immediately after unimplemented opcodes or the TRAP instruction, since the interrupt routine for these operations must have time to execute a PUSHF, PUSHA or DI. The SIGND prefix and the associated multiply or divide instructions must not be separated, so interrupts cannot occur after the SIGND opcode.

Setting and clearing the IPEND and IPEND1 registers is simplified since new interrupts are stored in buffer registers while read-modify-write operations are performed on IPEND and IPEND1. To set and clear bits in the pending registers the following sequences can be used:

ANDB IPEND, #11110111B; Clear IPEND.3

ORB IPEND, #00000010B; Set IPEND.1

The 80C196KA interrupt response time has been improved as follows:

States	80C196KA	
	External Stack	Internal Stack
	16	16
80C196KA @ 8 MHz	4.5	4.00
80C196KA @ 10 MHz	3.6	3.2

Interrupt Response Time

Interrupt response time is measured as the elapsed time from the end of the previous instruction to the beginning of the first instruction of the interrupt service routine. It does not include the time needed to finish the current instruction or to save values on the stack.

2.6 INSTRUCTION SET AND PSW

All the instructions in the 8096 exist in the 80C196KA and perform the same function with two exceptions. First, the PSW bits are set in a specific manner for some operations where the 8096 PSW results were undefined. Second, some instructions execute in fewer state times.

PSW Settings

The PSW bits on the 80C196KA are set as follows:

PSW:	7	6	5	4	3	2	1	0
	Z	N	V	VT	C	X	I	ST

Z: The Zero flag is set to indicate that an operation generated a result equal to zero. The instructions SUBC(B) and ADDC(B) can only clear the Z flag but can not set it. This makes it easier to perform double word arithmetic, as a zero in the high word will not set the zero flag.

N: The Negative flag is set to indicate that the operation generated a negative result. Note that the N flag will be in the algebraically correct state even if an overflow occurs. For shift operations, including the normalize operation and all three forms (SHL, SHR, SHRA) of byte, word and double word shifts, the N flag will be set to the same value as the most significant bit of the result. This will be true even if the shift count is 0.

V: The oVerflow flag is set to indicate that the operation generated a result which is outside the range for the destination data type. For divide operations, the following conditions are set:

For the operation: V is set if Quotient is:

UNSIGNED
BYTE DIVIDE > 255 (0FFH)

UNSIGNED
WORD DIVIDE > 65535 (0FFFFH)

SIGNED < -127 (81H)
or
BYTE
DIVIDE > 127 (7FH)

SIGNED < -32767 (8001H)
or
WORD
DIVIDE > 32767 (7FFFH)

VT: The oVerflow Trap flag is set when the V flag is set, but it is only cleared by the CLRVT, JVT and JNVT instructions. This allows testing for overflows in a group of operations instead of after each operation.

- C: The Carry flag is set to indicate the state of the arithmetic carry from the most significant bit of the ALU for an arithmetic operation, or the state of the last bit shifted out of an operand for a shift. Arithmetic Borrow after a subtract operation is the complement of the C flag (i.e. if the operation generated a borrow then C=0.)
- X: Reserved for future. Should always be cleared when writing to the PSW for compatibility with future products.
- I: The global Interrupt disable bit disables all interrupts except NMI when cleared.
- ST: The STicky bit is set to indicate that during a right shift a one has been shifted into the Carry flag and then has been shifted out. This flag can be used with the carry flag to determine rounding.

Descriptions of these new instructions follow:

1. **PUSHA** (push all): This instruction is used instead of PUSHF to support the 8 additional interrupts. It is similar to PUSHF, but pushes two words instead of one. The first word pushed is the same as for the PUSHF instruction, PSW/INT_MASK. The second word pushed is formed by the IMASK1/WSR register pair. As a result of this instruction the PSW, INT_MASK, and IMASK1 registers are cleared, and the SP is decremented by 4. Interrupts are disabled in two ways by this instruction since both PSW.9 and the interrupt masks are cleared. Interrupts cannot occur between this instruction and the one following it.

```

execution: SP ← SP - 2
           (SP) ← PSW/INT_MASK
           PSW/INT_MASK ← 0
           SP ← SP - 2
           (SP) ← IMASK1/WSR
           IMASK1 ← 0

```

```

assembly language format: PUSHA
object code format: <11110100>

```

```

bytes: 1
states: on-chip stack:12
       off-chip stack:18

```

PSW:	Z	N	V	VT	C	X	I	ST
	0	0	0	0	0	X	0	0

2. **POPA** (pop all): This instruction is used instead of POPF to support the 8 additional interrupts. It is similar to POPF, but pops two words instead of one. The first word is popped into the IMASK1/WSR register pair, while the second word is popped into the PSW/INT_MASK register pair. As a result of this instruction the SP is incremented by 4. Interrupts can not occur between this instruction and the one following it.

```

execution: IMASK1/WSR ← (SP)
           SP ← SP + 2
           PSW/INT_MASK ← (SP)
           SP ← SP + 2

```

```

assembly language format: POPA
object code format: <11110101>

```

Instruction Set Additions

Six instructions have been added to the 8096 instruction set to form the 80C196KA instruction set. The added instructions are:

- PUSHA** — PUSHes the PSW, INT_MASK, IMASK1, and WSR
- POPA** — POPs the PSW, INT_MASK, IMASK1, and WSR
- IDLPD** — Sets the part into IDLE or Powerdown mode
- DJNZW** — Decrement Jump Not Zero using a Word counter
- CMPL** — Compare 2 long direct values
- BMOV** — Block move using 2 auto-incrementing pointers and a counter

bytes: 1
states: on-chip stack: 12
off-chip stack: 18

PSW:

Z	N	V	VT	C	x	I	ST
✓	✓	✓	✓	✓	x	✓	✓

(✓ = changed)

3. **IDLPD** (idle/powerdown): This instruction is used for entry into the idle and powerdown modes. Selecting IDLE or POWERDOWN is done using the key operand. If the operand is not a legal key, the part executes a reset sequence. The bus controller will complete any prefetch cycle in progress before the CPU stops or resets.

execution: if KEY = 1 then enter IDLE
else if KEY = 2 then enter POWERDOWN
else execute reset.

assembly language format: IDLPD #key (key is 8-bit value)
object code format: <11110110> <key>

bytes: 2
states: legal key: 8
illegal key: 25

PSW:

	Z	N	V	VT	C	x	I	ST
Legal Key	-	-	-	-	-	x	-	-
Illegal Key	0	0	0	0	0	x	0	0

(- = Unchanged)

4. **DJNZW** (decrement and jump if not zero word): This instruction is the same as the DJNZ except that the count is a word operand. A counter word is decremented; if the result is not zero the jump is taken. The range of the jump is -128 to +127.

execution: COUNT ← COUNT - 1
if COUNT <> 0 then
PC ← PC + disp (sign extended)

assembly language format: DJNZW wreg, cadd
object code format: <11100001> <wreg> <disp>

bytes: 3
states: jump not taken: 5
jump taken: 9

PSW:

Z	N	V	VT	C	x	I	ST
-	-	-	-	-	x	-	-

5. **CMPL** (compare long): This instruction is used to compare the magnitudes of two double word (long) operands. The operands are specified using the direct addressing mode. Five PSW flags are set following this operation, but the operands are not affected.

execution: DST - SRC

DST SRC

assembly language format: CMPL Lreg,Lreg

object code format: <11000101> <src Lreg> <dst Lreg>

bytes: 3

states: 7

PSW:	Z	N	V	VT	C	x	I	ST
	✓	✓	✓	✓	✓	x	-	-

6. **BMOV** (block move): This instruction is used to move a block of word data from one location in memory to another. The source and destination addresses are calculated using the indirect with auto-increment addressing modes. A long register addresses the source and destination pointers which are stored in adjacent word registers. The number of transfers is specified by a word register. The blocks of data can reside anywhere in memory but should not overlap.

execution: COUNT ← (CNTREG)
 LOOP: SRCPTR ← (PTRS)
 DSTPTR ← (PTRS + 2)
 (DSTPTR) ← (SRCPTR)
 (PTRS) ← SRCPTR + 2
 (PTRS + 2) ← DSTPTR + 2
 COUNT ← COUNT - 1
 if COUNT <> 0 then go to LOOP

PTRS CNTREG

assembly language format: BMOV Lreg,wreg

object code format: <11000001> <wreg> <Lreg>

bytes: 3

states: internal/internal: 8 per transfer + 6

external/internal: 11 per transfer + 6

external/external: 14 per transfer + 6

PSW:	Z	N	V	VT	C	x	I	ST
	-	-	-	-	-	x	-	-

Notes:

1. CNTREG does not get decremented during the instruction
2. It is easy to unintentionally create a very long un-interruptable operation with this instruction.

To provide an interruptable version of BLKMOV for large blocks, the BLKMOV instruction can be used with the DJNZ(W) instruction. This is possible because the pointers are modified, but CNTREG is not. Consider the example:

```
LD PTRS, SRC ;Pointer to base of sources table
LD PTRS+2, DST ;Pointer to base of destination table
LD CNTREG, #COUNT;Number of words to move per set
LD CNTSET, #SETS ;Number of sets to move
BMOV PTRS, CNTREG ;Move one set
DJNZW CNTSET, MOVE ;Decrement set counters and move again
```

Addressing Modes

The instructions on the 80C196KA can be divided into 4 groups: no operand, one operand, two operand, and three operand. Two and three operand instructions, as well as the PUSH and POP instructions, can use multiple addressing modes, the remaining instructions can operate on any of the bytes in the register file or SFR space.

To indicate the address range for the operands of each instruction the letters "D", "B", and "A" are used. "D" is the destination register and must be in the register file or SFR space. "A" is the second operand. It is addressed using one of the six addressing modes and can be located anywhere in memory. "B" is the third operand for three operand instructions and must be located in the register file or SFR space. Three operand instructions reduce the number of temporary variables needed and therefore the number of move operations, speeding up the code for many applications.

The address modes usable with "A" operands are listed below:

Direct - The operand is specified by an 8-bit address field in the instruction. The operand must be in the Register File or SFR space.

Immediate - The operand itself follows the opcode in the instruction stream as immediate data. The immediate data can be either 8 or 16 bits wide.

Indirect - An 8-bit address field in the instruction contains the 7-bit address of a word in the Register File which contains the 16-bit address of the operand. The operand can be anywhere in memory

Indirect With Auto-Increment - Same as indirect, except that after the operand is referenced, the word register which contained its address is incremented by one if the operand is a byte or by two if it is a word.

Indexed (Long and Short) - The instruction contains an 8-bit address field and either an 8-bit or 16-bit displacement field. The 8-bit address field gives the 7-bit address of a word in the Register File which contains a 16-bit base address. The 8-bit or 16-bit displacement field contains a signed displacement which is added to the base address to produce the address of the operand. The operand can be anywhere in memory.

NOTE:

The indexed address mode can be used with the Zero Register to directly address any location in memory. It can also be used with the Stack Pointer to address variables on the stack.

The indexed and indirect modes of addressing on the 80C196KA operate in fewer state times than they do on the 8096 because of the extra 16-bit internal bus.

Figures 16 and 17 show a summary of the instructions available on the 80C196KA and the number of state times each requires to execute. Timing values for jumps, calls and returns include the time required to flush the instruction queue and to fetch the opcode at the destination address.

The instruction times listed are the minimum number of state times required for execution. (A state time is 2 oscillator periods.) This number could increase if wait states are used or if the opcode and its operands are not prefetched and residing in the instruction queue when they are needed. The instruction queue is almost never empty when running in the 16-bit bus mode without wait states, so the minimum number of state times is almost always the correct execution time.

As would be expected, some performance degradation occurs when using wait states or the 8-bit bus since the queue may become empty. It is very difficult to predict the exact queue status at all times, so the instruction timings can not be exactly predicted, only minimum and worst case timings can be calculated.

When adding wait-states, the number of wait-states used, multiplied by the number of instruction fetches and data accesses occurring, must be added to the instruction execution timing. This will provide the worst-case timing for an instruction sequence, the actual timing will be between the minimum timing and the worst-case timing.

In the 8-bit bus mode, the worst case timing, assuming no wait-states, can be calculated by adding the following to the minimum timings:

2 state times for each external word write

1 state time for each external word read

1 state time for each byte that is not in the queue when needed (worst case is the number of bytes in an instruction minus 1)

Instruction execution in the 8-bit mode typically takes 20 to 30 percent longer than in the 16-bit mode.

Instruction Summary

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	$D \leftarrow D + A$	✓	✓	✓	✓	↑	—	
ADD/ADDB	3	$D \leftarrow B + A$	✓	✓	✓	✓	↑	—	
ADDC/ADDCB	2	$D \leftarrow D + A + C$	↓	✓	✓	✓	↑	—	
SUB/SUBB	2	$D \leftarrow D - A$	✓	✓	✓	✓	↑	—	
SUB/SUBB	3	$D \leftarrow B - A$	✓	✓	✓	✓	↑	—	
SUBC/SUBCB	2	$D \leftarrow D - A + C - 1$	↓	✓	✓	✓	↑	—	
CMP/CMPB	2	$D - A$	✓	✓	✓	✓	↑	—	
MUL/MULU	2	$D, D + 2 \leftarrow D \times A$	—	—	—	—	—	—	2
MUL/MULU	3	$D, D + 2 \leftarrow B \times A$	—	—	—	—	—	—	2
MULB/MULUB	2	$D, D + 1 \leftarrow D \times A$	—	—	—	—	—	—	3
MULB/MULUB	3	$D, D + 1 \leftarrow B \times A$	—	—	—	—	—	—	3
DIVU	2	$D \leftarrow (D, D + 2) / A, D + 2 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	2
DIVUB	2	$D \leftarrow (D, D + 1) / A, D + 1 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	3
DIV	2	$D \leftarrow (D, D + 2) / A, D + 2 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	
DIVB	2	$D \leftarrow (D, D + 1) / A, D + 1 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	
AND/ANDB	2	$D \leftarrow D \text{ AND } A$	✓	✓	0	0	—	—	
AND/ANDB	3	$D \leftarrow B \text{ AND } A$	✓	✓	0	0	—	—	
OR/ORB	2	$D \leftarrow D \text{ OR } A$	✓	✓	0	0	—	—	
XOR/XORB	2	$D \leftarrow D \text{ (excl. or) } A$	✓	✓	0	0	—	—	
LD/LDB	2	$D \leftarrow A$	—	—	—	—	—	—	
ST/STB	2	$A \leftarrow D$	—	—	—	—	—	—	
LDBSE	2	$D \leftarrow A; D + 1 \leftarrow \text{SIGN}(A)$	—	—	—	—	—	—	3,4
LDBZE	2	$D \leftarrow A; D + 1 \leftarrow 0$	—	—	—	—	—	—	3,4
PUSH	1	$SP \leftarrow SP - 2; (SP) \leftarrow A$	—	—	—	—	—	—	
POP	1	$A \leftarrow (SP); SP + 2$	—	—	—	—	—	—	
PUSHF	0	$SP \leftarrow SP - 2; (SP) \leftarrow \text{PSW};$ $\text{PSW} \leftarrow 0000\text{H}; I \leftarrow 0$	0	0	0	0	0	0	
POPF	0	$\text{PSW} \leftarrow (SP); SP \leftarrow SP + 2; I \leftarrow \text{✓}$	✓	✓	✓	✓	✓	✓	
SJMP	1	$PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LJMP	1	$PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5
BR[indirect]	1	$PC \leftarrow (A)$	—	—	—	—	—	—	
SCALL	1	$SP \leftarrow SP - 2;$ $(SP) \leftarrow PC; PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LCALL	1	$SP \leftarrow SP - 2; (SP) \leftarrow PC;$ $PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5

Figure 16. Instruction Summary

Instruction Summary (Continued)

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
RET	0	PC ← (SP); SP ← SP + 2	—	—	—	—	—	—	
J (conditional)	1	PC ← PC + 8-bit offset (if taken)	—	—	—	—	—	—	5
JC	1	Jump if C = 1	—	—	—	—	—	—	5
JNC	1	jump if C = 0	—	—	—	—	—	—	5
JE	1	jump if Z = 1	—	—	—	—	—	—	5
JNE	1	Jump if Z = 0	—	—	—	—	—	—	5
JGE	1	Jump if N = 0	—	—	—	—	—	—	5
JLT	1	Jump if N = 1	—	—	—	—	—	—	5
JGT	1	Jump if N = 0 and Z = 0	—	—	—	—	—	—	5
JLE	1	Jump if N = 1 or Z = 1	—	—	—	—	—	—	5
JH	1	Jump if C = 1 and Z = 0	—	—	—	—	—	—	5
JNH	1	Jump if C = 0 or Z = 1	—	—	—	—	—	—	5
JV	1	Jump if V = 0	—	—	—	—	—	—	5
JNV	1	Jump if V = 1	—	—	—	—	—	—	5
JVT	1	Jump if VT = 1; Clear VT	—	—	—	—	0	—	5
JNVT	1	Jump if VT = 0; Clear VT	—	—	—	—	0	—	5
JST	1	Jump if ST = 1	—	—	—	—	—	—	5
JNST	1	Jump if ST = 0	—	—	—	—	—	—	5
JBS	3	Jump if Specified Bit = 1	—	—	—	—	—	—	5,6
JBC	3	Jump if Specified Bit = 0	—	—	—	—	—	—	5,6
DJNZ/ DJNZW	1	D ← D - 1; If D ≠ 0 then PC ← PC + 8-bit offset	—	—	—	—	—	—	5
DEC/DECB	1	D ← D - 1	✓	✓	✓	✓	↑	—	
NEG/NEGB	1	D ← 0 - D	✓	✓	✓	✓	↑	—	
INC/INCB	1	D ← D + 1	✓	✓	✓	✓	↑	—	
EXT	1	D ← D; D + 2 ← Sign (D)	✓	✓	0	0	—	—	2
EXTB	1	D ← D; D + 1 ← Sign (D)	✓	✓	0	0	—	—	3
NOT/NOTB	1	D ← Logical Not (D)	✓	✓	0	0	—	—	
CLR/CLRB	1	D ← 0	1	0	0	0	—	—	
SHL/SHLB/SHLL	2	C ← msb ----- lsb ← 0	✓	✓	✓	✓	↑	—	7
SHR/SHRB/SHRL	2	0 → msb ----- lsb → C	✓	✓	✓	0	—	✓	7
SHRA/SHRAB/SHRAL	2	msb → msb ----- lsb → C	✓	✓	✓	0	—	✓	7
SETC	0	C ← 1	—	—	1	—	—	—	
CLRC	0	C ← 0	—	—	0	—	—	—	

Figure 16. Instruction Summary (Continued)

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
CLRVT	0	VT ← 0	—	—	—	0	0	—	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	—	—	—	—	—	—	
EI	0	Enable All Interrupts (I ← 1)	—	—	—	—	—	—	
NOP	0	PC ← PC + 1	—	—	—	—	—	—	
SKIP	0	PC ← PC + 2	—	—	—	—	—	—	
NORML	2	Left shift till msb = 1; D ← shift count	✓	✓	0	—	—	—	7
TRAP	0	SP ← SP - 2; (SP) ← PC; PC ← (2010H)	—	—	—	—	—	—	9
PUSHA	1	SP ← SP-2; (SP) ← PSW; PSW ← 0000H; SP ← SP-2; (SP) ← IMASK1/WSR; IMASK1 ← 00H	0	0	0	0	0	0	
POPA	1	IMASK1/WSR ← (SP); SP ← SP+2 PSW ← (SP); SP ← SP+2	✓	✓	✓	✓	✓	✓	
IDLDP	1	IDLE MODE IF KEY = 1; POWERDOWN MODE IF KEY = 2; CHIP RESET OTHERWISE	—	—	—	—	—	—	
CMPL	2	D-A	✓	✓	✓	✓	↑	—	
BMOV	2	[PTR_HI] + ← [PTR_LOW] + ; UNTIL COUNT = 0	—	—	—	—	—	—	

NOTES:

1. If the mnemonic ends in "B" a byte operation is performed, otherwise a word operation is done. Operands is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the Register File; A can be located anywhere in memory.
2. D, D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D, D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to word.
5. Offset is a 2's complement number.
6. Specified bit is one of the 2048 bits in the register file.
7. The "L" (Long) suffix indicates double-word operation.
8. Initiates a Reset by pulling RESET low. Software should re-initialize all the necessary registers with code starting at 2080H.
9. The assembler will not accept this mnemonic.

Figure 16. Instruction Summary (Continued)

3	—	0	✓	✓	✓	✓	✓	✓	EXT
3	—	0	✓	✓	✓	✓	✓	✓	EXTB
3	—	0	✓	✓	✓	✓	✓	✓	NOT/NOTB
3	—	0	✓	✓	✓	✓	✓	✓	CLR/CLRB
3	—	0	✓	✓	✓	✓	✓	✓	SHL/SHLB/SHL
3	—	0	✓	✓	✓	✓	✓	✓	SHR/SHRB/SHR
3	—	0	✓	✓	✓	✓	✓	✓	SHRA/SHRB/SHRAL
3	—	0	✓	✓	✓	✓	✓	✓	RET
3	—	0	✓	✓	✓	✓	✓	✓	CLRC

MNEMONIC	DIRECT	IMMED	INDIRECT		INDEXED	
			NORMAL*	A-INC*	SHORT*	LONG*
ADD (3-op)	5	6	7/9	8/10	7/9	8/10
SUB (3-op)	5	6	7/9	8/10	7/9	8/10
ADD (2-op)	4	5	6/8	7/9	6/8	7/9
SUB (2-op)	4	5	6/8	7/9	6/8	7/9
ADDC	4	5	6/8	7/9	6/8	7/9
SUBC	4	5	6/8	7/9	6/8	7/9
CMP	4	5	6/8	7/9	6/8	7/9
ADDB (3-op)	5	5	7/9	8/10	7/9	8/10
SUBB (3-op)	5	5	7/9	8/10	7/9	8/10
ADDB (2-op)	4	4	6/8	7/9	6/8	7/9
SUBB (2-op)	4	4	6/8	7/9	6/8	7/9
ADDCB	4	4	6/8	7/9	6/8	7/9
SUBCB	4	4	6/8	7/9	6/8	7/9
CMPB	4	4	6/8	7/9	6/8	7/9
MUL (3-op)	16	17	18/21	19/22	19/22	20/23
MULU (3-op)	14	15	16/19	17/20	17/20	18/21
MUL (2-op)	16	17	18/21	19/22	19/22	20/23
MULU (2-op)	14	15	16/19	17/20	17/20	18/21
DIV	26	27	28/31	29/32	29/32	30/33
DIVU	24	25	26/29	27/30	27/30	28/31
MULB (3-op)	12	12	14/17	15/18	15/18	16/19
MULUB (3-op)	10	10	12/15	12/16	12/16	14/17
MULB (2-op)	12	12	14/17	15/18	15/18	16/19
MULUB (2-op)	10	10	12/15	12/16	12/16	14/17
DIVB	18	18	20/23	21/24	21/24	22/25
DIVUB	16	16	18/21	19/22	19/22	20/23
AND (3-op)	5	6	7/9	8/10	7/9	8/10
AND (2-op)	4	5	6/8	7/9	6/8	7/9
OR (2-op)	4	5	6/8	7/9	6/8	7/9
XOR	4	5	6/8	7/9	6/8	7/9
ANDB (3-op)	5	5	7/9	8/10	7/9	8/10
ANDB (2-op)	4	4	6/8	7/9	6/8	7/9
ORB (2-op)	4	4	6/8	7/9	6/8	7/9
XORB	4	4	6/8	7/9	6/8	7/9
LD/LDB	4	5	5/7	6/8	6/8	7/9
ST/STB	4	5	5/7	6/8	6/8	7/9
LDBSE	4	4	5/7	6/8	6/8	7/9
LDBZE	4	4	5/7	6/8	6/8	7/9
BMOV		6 + 8 per word		6 + 11/14 per word		
PUSH (int stack)	6	7	9/12	10/13	10/13	11/14
POP (int stack)	8	—	10/12	11/13	11/13	12/14
PUSH (ext stack)	8	9	11/14	12/15	12/15	13/16
POP (ext stack)	11	—	13/15	14/16	14/16	15/17

*Times for (Internal/External) Operands

Figure 17a. Instruction Execution State Times

MNEMONIC	INDIRECT	MNEMONIC	MNEMONIC
PUSHF (int stack)	6	PUSHF (ext stack)	8
POPF (int stack)	7	POPF (ext stack)	10
PUSHA (int stack)	12	PUSHA (ext stack)	18
POPA (int stack)	12	POPA (ext stack)	18
TRAP (int stack)	16	TRAP (ext stack)	18
LCALL (int stack)	11	LCALL (ext stack)	13
SCALL (int stack)	11	SCALL (ext stack)	13
RET (int stack)	11	RET (ext stack)	14
CMPL	7	DEC/DECB	3
CLR/CLRB	3	EXT/EXTB	4
NOT/NOTB	3	INC/INCB	3
NEG/NEGB	3		
LJMP	7		
SJMP	7		
BR [indirect]	7		
JNST, JST	4/8 jump not taken/jump taken		
JNH, JH	4/8 jump not taken/jump taken		
JGT, JLE	4/8 jump not taken/jump taken		
JNC, JC	4/8 jump not taken/jump taken		
JNVT, JVT	4/8 jump not taken/jump taken		
JNV, JV	4/8 jump not taken/jump taken		
JGE, JLT	4/8 jump not taken/jump taken		
JNE, JE	4/8 jump not taken/jump taken		
JBC, JBS	5/9 jump not taken/jump taken		
DJNZ	5/9 jump not taken/jump taken		
DJNZW	5/9 jump not taken/jump taken		
NORML	8 + 1 per shift (9 for 0 shift)		
SHRL	7 + 1 per shift (8 for 0 shift)		
SHLL	7 + 1 per shift (8 for 0 shift)		
SHRAL	7 + 1 per shift (8 for 0 shift)		
SHR/SHRB	6 + 1 per shift (7 for 0 shift)		
SHL/SHLB	6 + 1 per shift (7 for 0 shift)		
SHRA/SHRAB	6 + 1 per shift (7 for 0 shift)		
CLRC	2		
SETC	2		
DI	2		
EI	2		
CLRVT	2		
NOP	2		
RST	15 (includes fetch of configuration byte)		
SKIP	3		
IDLPD	8/25 (proper key/improper key)		

Figure 17b. Instruction Execution State Times

3.0 PERIPHERAL DESCRIPTION

3.1 OVERVIEW

There are five major peripherals on the 80C196KA: the serial port, analog to digital converter, pulse-width-modulated output, standard I/O ports and the high speed I/O unit. With the exception of the high speed I/O unit (HSIO), each of the peripherals is a single unit that can be discussed without further separation. These peripherals will be described after the HSIO unit.

Four individual sections make up the HSIO and work together to form a very flexible timer/counter based I/O system. Included in the HSIO are a 16-bit timer (TIMER1), a 16-bit up/down counter (TIMER2), a programmable high speed input unit (HSI), and a programmable high speed output unit (HSO).

With very little CPU overhead the HSIO can measure pulse widths, generate waveforms, and create periodic interrupts. Depending on the application, it can perform the work of up to 18 timer/counters and capture/compare registers. Timer1 and Timer2 are used as the time bases for the HSIO. After describing their operation, the HSI and then the HSO will be discussed.

3.2 TIMERS

Timer1

Timer1 is a free-running timer which is incremented every eight state times. It can be read and written, but care must be taken when writing to it if the High Speed I/O (HSIO) Subsystem is being used. The precautions necessary when writing to Timer1 are described in the HSIO section. Timer1 can cause an interrupt when it overflows from 0FFFFH to 0000H if enabled by setting IOC1.2 = 1.

Timer2

Timer2 on the 80C196KA has many enhancements over Timer2 on the 8096. It counts transitions, both positive and negative, on its input which can be either the T2CLK pin or the HSI.1 pin depending on the state of IOC0.7. The maximum transition speed is once per state time in the Fast Increment mode, and once every 8 states otherwise. Timer2 can be read and written and can be reset by hardware, software or the HSO unit.

Interrupts can be generated if Timer2 crosses the 0FFFFH/0000H boundary or the 7FFFH/8000H boundary in either direction. By having two interrupt points it is possible to have interrupts enabled even if Timer2 is counting up and down centered around one of the interrupt points. The interrupt can be set to vector through location 2038H or 2000H using the interrupt mask registers and IOC1.3.

The value in Timer2 can be captured into the T2CAPTURE register by a rising edge on P2.7. T2CAP is located at 0CH in register plane 15. The interrupt generated by a capture vectors through location 2036H.

Timer2 can be placed in the Fast Increment mode by setting IOC2.0. In this mode it is not synchronized to the HSO unit and may not work properly with the HSO if transitions occur faster than every 8 states. In addition, HSO events based on Timer2 may not occur as expected if a count transition occurs within 8 state times before or after the timer is reset by other than an HSO event.

Timer2 can be made to count up or down based on the Port 2.6 pin if IOC2.1 = 1. However, caution must be used when this feature is working in conjunction with the HSO. If Timer2 does not complete a full cycle it is possible to have events in the CAM which never match the timer. These events would stay in the CAM until the CAM is cleared or the chip is reset.

The following control/status bits are associated with the Timer2:

	Bit = 1	Bit = 0
IOC0.1	Reset Timer2 each write	No action
IOC0.3	Enable external reset	Disable
IOC0.5	HSI.0 is ext. reset source	T2RST is reset source
IOC0.7	HSI.1 is T2 clock source	T2CLK is clock source
IOC1.3	Enable Timer2 overflow int.	Disable overflow interrupt
IOC2.0	Enable fast increment	Disable fast increment
IOC2.1	Enable downcount feature	Disable downcount
P2.6	Count down if IOC2.1 = 1	Count up
IOC2.5	Interrupt on 7FFFH/8000H	Interrupt on 0FFFFH/0000H
P2.7	Capture Timer2 into T2CAPTURE on rising edge	

int_1

The High Speed Input (HSI) unit can capture the value of Timer1 when an event takes place on one of four input lines. Four types of events can trigger a capture; rising edges only, falling edges only, rising or falling edges or every eighth rising edge. Whenever the every eighth rising edge mode is entered the divide-by-8 counter is reset, allowing very fast pulses to be measured and counted. The input lines are sampled for events during every Phase1. A block diagram of this unit is shown in Figure 18.

Each of the input lines can be individually programmed to select the type of event to trigger on using the HSI_MODE register (shown in Figure 19). Several bits of the IOC0 register enable and disable the HSI lines, as well as control the inputs to Timer2. The function of these bits is shown in Figure 20.

When events occur, the Timer1 value and 4 status bits indicating which line(s) had events get stored in a 7 level fifo. The next event ready to be unloaded from the fifo is placed in the HSI Holding Register, so a total of 8 pieces of data can be stored in the fifo. If events occur after the fifo is full they will not be recorded and the fifo will contain the information gathered prior to the overflow error condition.

register, followed by reading the HSI_TIME register. When the high byte of the time register is read the next fifo location is loaded into the holding register, so reading HSI_TIME before HSI_STATUS will result in getting the wrong status information. For convenience the HSI time register should be read as a word. The HSI unit is synchronized to Timer1 which increments every 8 state times. For this reason it is required that 8 state times elapse between reading HSI_TIME and the next HSI_STATUS. The HSI_STATUS register, shown in Figure 21, also contains bits which indicate the level of the HSI pins at the time that HSI_STATUS is read.

The HSI can generate interrupts in three ways: each time a value moves from the fifo into the holding register; when the fifo (independent of the holding register) has 4 or more events stored; when the fifo has 6 or more events stored. The first case is called FIFO_LOADED, the second is FIFO_4, and the last case is called FIFO_FULL. Either the FIFO_LOADED or the FIFO_FULL interrupts can be selected by IOC1.7 to vector through location 2004H. The FIFO_4 interrupt vectors through location 2034H, and the FIFO_LOADED interrupt vectors through location 203CH. An additional interrupt can be generated by a rising edge on the HSI.0 pin, even if the pin is not enabled to the HSI unit. This interrupt vectors through location 2008H.

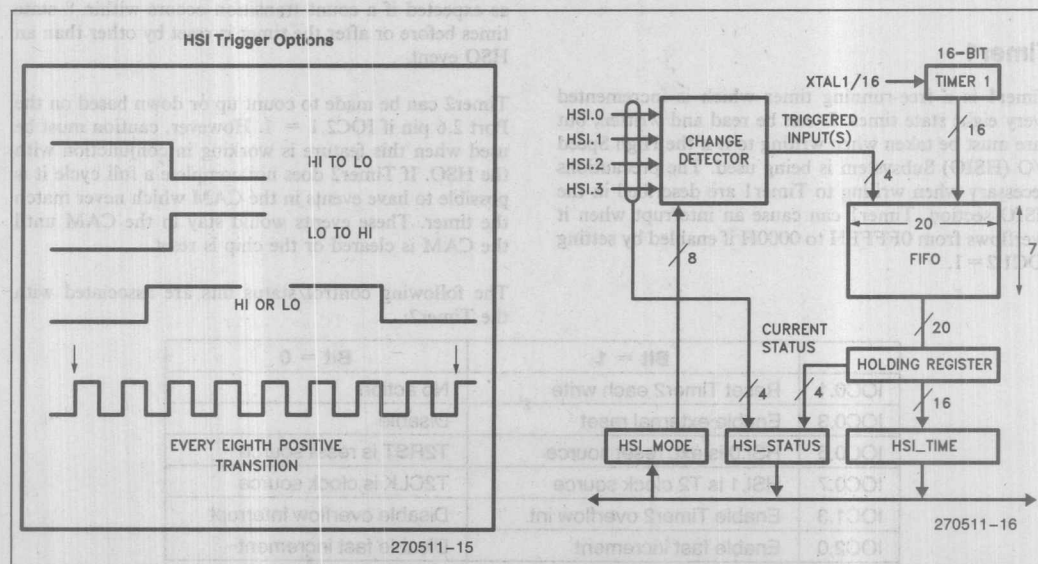


Figure 18. HSI Block Diagram

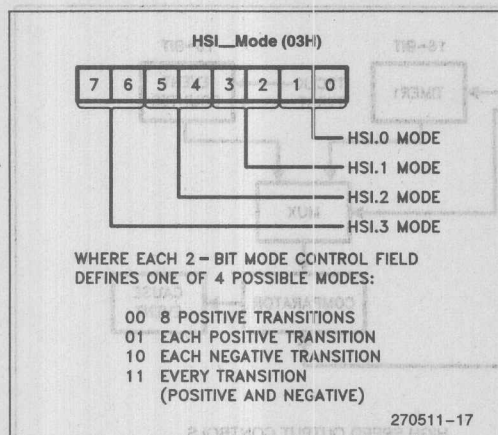


Figure 19. HSI Mode Register

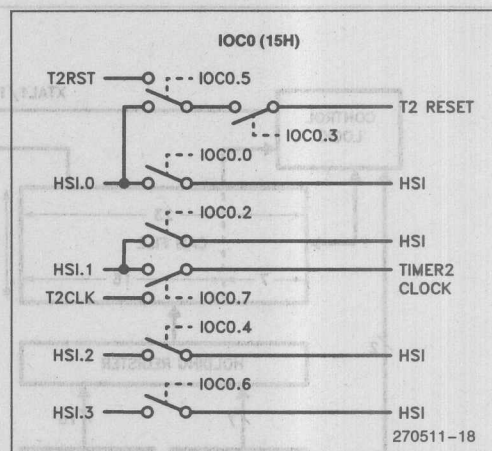


Figure 20. IOC0 Control of the HSI

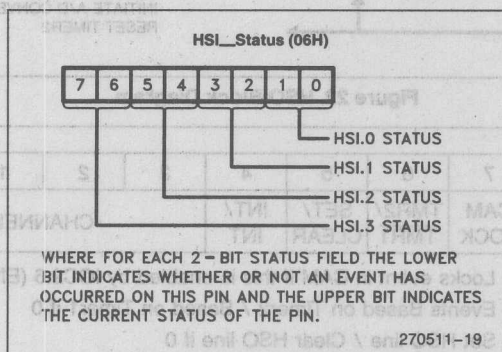


Figure 21. HSI Status Register

3.4 HIGH SPEED OUTPUTS (HSO)

The High Speed Output (HSO) unit can generate events at specified times or counts based on Timer1 or Timer2. A block diagram of the HSO unit is shown in Figure 22. Up to 8 pending events can be stored in the CAM (Content Addressable Memory) of the HSO unit at one time. Commands are placed into the HSO unit by first writing to HSO_COMMAND with the event to occur, and then to HSO_TIME with the timer match value. Although HSO_TIME is usually written as a word, it is the writing of the high byte which sends the command into the CAM. Since the HSO is synchronized to Timer1 and the HSI, 8 state times must elapse between writing to HSO_TIME and writing the next HSO_COMMAND.

Sixteen different types of events can be triggered by the HSO: 8 external and 8 internal. There are two interrupt vectors associated with the HSO. The one at 2006H is used for external events, the one at 200AH, called the Software Timer Interrupt, is used for internal events. External events consist of switching up to 6 lines, HSO.0 through HSO.5. These lines switch during Phase1. (Note that HSO.4 and HSO.5 are shared with HSI.2 and HSI.3.)

Internal events include setting up 4 Software Timers, resetting Timer2, and starting an A to D conversion. The software timers are flags that can be set by the HSO and optionally cause interrupts. The format for the HSO commands is shown in Figure 23. Note that commands 0C and 0D will act as additional software timer commands with no associated status bit. They are useful only if the interrupt bit (bit4) is set in the HSO_COMMAND.

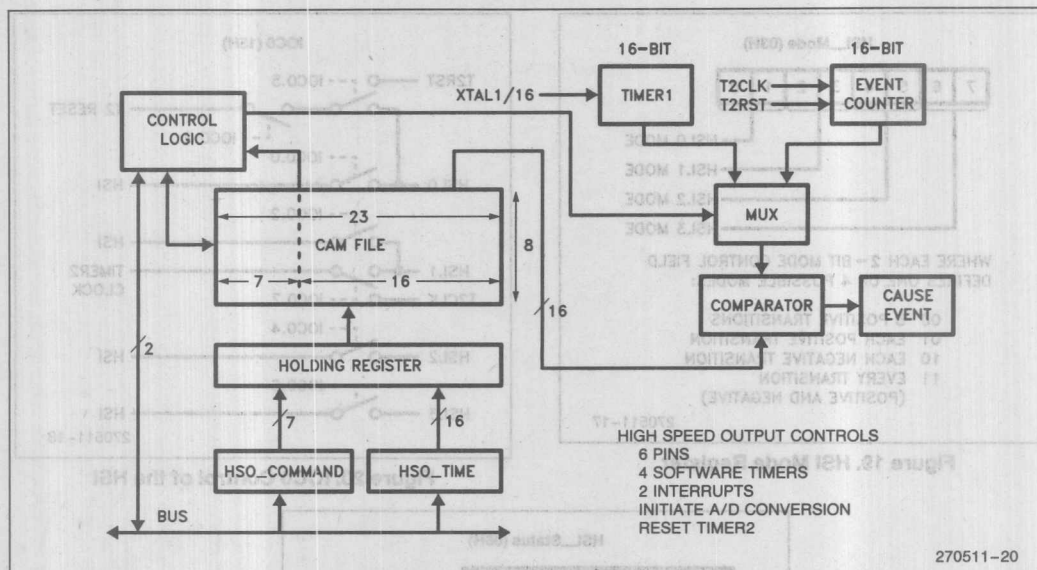


Figure 22. HSO Block Diagram

	7	6	5	4	3	2	1	1
HSO__ COMMAND	CAM LOCK	TMR2/ TMR1	SET/ CLEAR	INT/ INT	CHANNEL			

CAM Lock — Locks event in CAM if this is enabled by IOC2.6 (ENA_LOCK)

TMR/TMR1 — Events Based on Timer2 / Based on Timer1 if 0

SET/CLEAR — Set HSO line / Clear HSO line if 0

INT/INT — Cause Interrupt / No interrupt if 0

CHANNEL: 0-5: HSO lines 0-5

(in Hex): 6: HSO lines 0 and 1

7: HSO lines 2 and 3

8-B: Software Timers 0-4

C-D: Unflagged Events

E: Reset Timer2

F: Start A to D conversion

Figure 23. HSO Command Register

The CAM Lock bit (HSO__Command.7) can be set to keep commands in the CAM, otherwise the commands will clear from the CAM as soon as they cause an event. This feature is best used to generate periodic events based on Timer2 and must be enabled by setting IOC2.6. To clear locked events from the CAM, the entire CAM must be cleared by writing a one to the CAM clear bit IOC2.7. A chip reset will also clear the CAM. It is possible to cancel individual external events by writing the opposite event to the CAM and setting it to

occur at the same time. Both of these events will then remain in the CAM until the time tag is matched. Since HSO events are dependent on exact matches of the timers with the values in the CAM, it is important to be very careful when using timers in any mode except continuous counting in one direction. If Timer2 is used in the Fast Count mode, the HSO should not be used if counts could occur faster than once every 8 state times.

A status register, IOS2, has been added to the 80C196KA to indicate which events have been generated by the HSO unit. IOS2 is cleared whenever it is accessed (a jump on bit is considered an access). The correspondence between the HSO events and the bits in the IOS2 is shown below.

IOS2:	7	6	5	4	3	2	1	0
	HC15	HC14	HSO.5	HSO.4	HSO.3	HSO.2	HSO.1	HSO.0

Bits 0 through 5 indicate that a command affecting the corresponding HSO pin was executed. Bits 6 and 7 indicate occurrence of HSO_CMD_14 and HSO_CMD_15 respectively (Reset Timer2 and Start A/D Converter.) This register clears on read.

The IOS0 register contains the status of the HSO lines. When WSR = 15, writing to this register changes the values on the HSO pins. However, the HSO can change this written value by executing a command. The IOS0 register format is shown below.

IOS0:	7	6	5	4	3	2	1	0
	H.REG	CAM	HSO.5	HSO.4	HSO.3	HSO.2	HSO.1	HSO.0

Bits 0 through 5 indicate the state of the I/O line. Bits 6 and 7 indicate that a space is available in the CAM and a space is available in the holding register, respectively.

3.5 SERIAL PORT

The serial port on the 80C196KA has three full-duplex asynchronous modes and one synchronous mode. All of the modes are compatible with the other MCS®-96 parts and members of the MCS®-51 product family. The synchronous mode is called Mode 0, the asynchronous modes are called Modes 1, 2 and 3. An independent baud rate generator determines the baud rate for all of the modes. The baud rate value is different than that used for the 8096.

Mode 0

Mode 0 synchronous operation uses the RXD pin to input or output data 8 bits at a time. TXD is used to output the clock signal. The low time of the clock is always two states except in the fastest mode. In the fastest mode, set by entering a 8001H into the baud register, the low and high times of the clock are each one state time. Figure 24 shows the relative timings of the serial port operating in Mode 0.

Mode 1

Mode 1 is the standard asynchronous serial communication mode. A 10-bit frame (shown in Figure 25) is transmitted or received using a start bit, 8 data bits, and a stop bit. If parity is enabled by setting PEN = 1, an even parity bit is sent instead of the 8th data bit and parity is checked on reception.

Mode 2

Mode 2 is the 9th bit recognition mode and is frequently used with Mode 3 in interprocessor communication. In this mode an 11-bit frame (shown in Figure 25) consisting of a start bit, 9 data bits, and a stop bit are sent and received. When transmitting, the 9th bit can be set using TB8. During reception the RI flag and interrupts will not be set unless the 9th data bit is high. Parity cannot be enabled in this mode.

Mode 3

Mode 3 uses the same 11-bit frame as Mode 2. When transmitting, parity can be enabled, providing 8 data bits and an even parity bit in place of the 9th data bit. When receiving, the RI bit is always set and the RB8 bit contains the value of the 9th data bit. If parity is enabled, (PEN = 1), the RB8/RPE bit will indicate a parity error if one occurs.

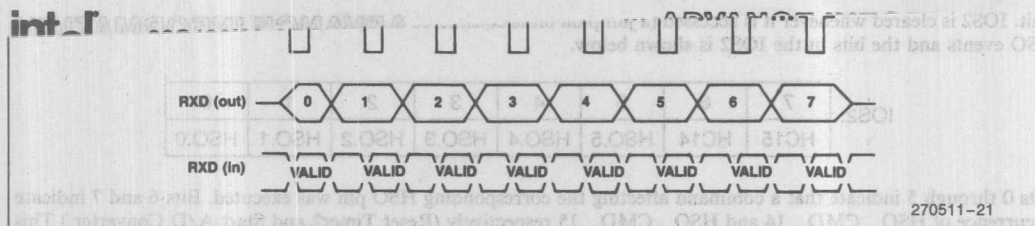


Figure 24. Serial Port Mode 0 Timings

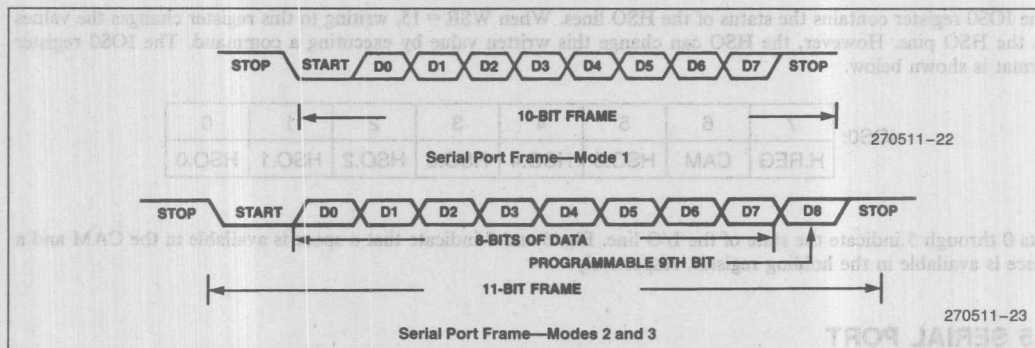


Figure 25. Serial Port Frames, Modes 1, 2 and 3

Baud Rates

Baud rates are generated based on either the T2CLK pin or XTAL1 pin. Baud rates are calculated using the following formulas where BAUD_REG is the value loaded into the baud rate register:

Asynchronous Modes 1, 2 and 3:

$$\text{BAUD_REG} = \frac{\text{XTAL1}}{\text{Baud Rate} \times 16} - 1 \quad \text{OR} \quad \frac{\text{T2CLK}}{\text{Baud Rate} \times 8}$$

Synchronous Mode 0:

$$\text{BAUD_REG} = \frac{\text{XTAL1}}{\text{Baud Rate} \times 2} - 1 \quad \text{OR} \quad \frac{\text{T2CLK}}{\text{Baud Rate}}$$

The most significant bit in the baud register value is set to a one to select XTAL1 as the source. If it is a zero

the T2CLK pin becomes the source. The following table shows some typical baud rate values:

BAUD RATES AND BAUD REGISTER VALUES

BAUD RATE	XTAL1 FREQUENCY		
	8.0 MHz	10.0 MHz	12.0 MHz
300	1666 / -0.02	2082 / 0.02	2499 / 0.00
1200	416 / -0.08	520 / -0.03	624 / 0.00
2400	207 / 0.16	259 / 0.16	312 / -0.16
4800	103 / 0.16	129 / 0.16	155 / 0.16
9600	51 / 0.16	64 / 0.16	77 / 0.16
19.2K	25 / 0.16	32 / 1.40	38 / 0.16

Baud Register Value / % error

A maximum baud rate of 750 Kbaud is available in the asynchronous modes with 12MHz on XTAL1. The synchronous mode has a maximum rate of 3.0 Mbaud with a 12 MHz clock. Location 0EH is the Baud Register. It is loaded sequentially in two bytes, with the low byte being loaded first. This register may not be loaded with zero in serial port Mode 0.

Serial Port Control

Reading the serial port is done through the Serial Buffer receive (SBUF(RX)) register at location 7. This register is double buffered so data can continually be received. Writing to the serial port is done through SBUF(TX), also addressed at location 7. This register is double buffered on the 80C196KA to allow two bytes at a time to be written to the serial port.

Serial port control is done through the Serial Port Control (SP_CON) register at location 11H. This register is write-only in Window 0 and has the following format:

SP_CON:	7	6	5	4	3	2	1	0
	X	X	X	TB8	REN	PEN	M2	M1

- TB8 — Sets the ninth data bit for transmission. Cleared after each transmission. Not valid if parity is enabled
- REN — Enables the receiver
- PEN — Enables the Parity function (even parity)
- M2,M1 — Sets the mode. Mode0=00, Mode1=01, Mode2=10, Mode3=11

The status of the serial port is read through the bits in the Serial Port Status (SP_STAT) register, also at location 11H. Figure 21 shows the status bits of this register. On the 80C196KA the SP_STAT register contains new bits to indicate receive Overrun Error (OE), Framing Error (FE), and Transmitter Empty (TXE). The bits which were also present on the 8096 are the Transmit Interrupt (TI) bit, the Receive Interrupt (RI) bit, and the Received Bit 8 (RB8) or Receive Parity Error (RPE) bit. SP_STAT is read-only in Window 0 and has the following format:

SP_STAT	7	6	5	4	3	2	1	0
	RB8/ RPE	RI	TI	FE	TXE	OE	X	X

- RB8 — Set if the 9th data bit is high on reception (parity disabled)
- RPE — Set if parity is enabled and a parity error occurred
- RI — Set at the end of the STOP bit reception
- TI — Set at the beginning of the STOP bit transmission
- FE — Set if no STOP bit is found at the end of a reception

- TXE — Set if two bytes can be sent to SBUF(TX)
- OE — Set if a byte is lost because SBUF was not read fast enough

The receiver on the 80C196KA checks for a valid stop bit. When one is detected, the data in the receive shift register is loaded into SBUF(RX). If a stop bit is not found within the appropriate time the Framing Error (FE) bit is set. In either case, the data in the receive shift register is loaded into SBUF(RX) and the RI bit is set. If this happens before the previous byte in SBUF(RX) is read, the Overflow Error (OE) bit is set. The data in SBUF(RX) will always be the latest byte received; it will never be a combination of the two bytes. When the RI bit is set it can cause an interrupt through the vectors at locations 200CH and 2032H. The RI, OE, and FE bits are reset when SP_STAT is read.

The Transmitter Empty (TXE) bit is set if the transmit FIFO is empty and ready to take up to two characters to be sent. TXE gets cleared as soon as a byte is written to SBUF. Two bytes may be written consecutively to SBUF if TXE is set. One byte may be written if TI alone is set. By definition, if TXE has just been set, a transmission has completed and TI will be set. When the TI bit is set it can cause an interrupt through the vectors at locations 200CH and 2032H. The user should not mask off this interrupt when using the double-buffered feature of the transmitter, as it could cause a missed count in the number of bytes being transmitted. The TI bit is reset when the CPU reads the SP_STAT registers.

3.6 A-TO-D CONVERTER

The 80C196KA A-to-D converter has 10 bits of resolution and can be run in modes compatible with either the 8096-90 or the 8096BH. Conversions can be performed on one of eight channels, the inputs of which share pins with port 0. The A to D includes a switchable Sample and Hold feature for the selected channel and does the conversion in as little as 91 state times.

Conversions are started by loading the AD_COMMAND register at location 02H with the channel number. The conversion can be started immediately by setting the GO bit to a one. If it is cleared the conversion will start when the HSO unit triggers it. The AD_COMMAND register has the following format:

A TO D_	7	6	5	4	3	2	1	0
COMMAND:	X	X	X	X	GO	CHANNEL NUMBER		

The A-to-D converter can cause an interrupt to occur through the vector at location 2002H when it completes a conversion. It is also possible to use a polling method by checking the Status (S) bit in the lower byte of the AD__RESULT register, also at location 02H. The status bit will be a 1 while a conversion is in progress. It takes 8 state times to set this bit after a conversion is started. The upper byte of the result register contains the most significant 8 bits of the conversion. The lower byte format is shown below:

A TO D__ RESULT__ LO:	7	6	5	4	3	2	1	0
	LOWEST 2 RESULT BITS		X	X	S	CHANNEL NUMBER		

At high crystal frequencies, more time is needed to allow the comparator to settle. For this reason IOC2.4 is provided to adjust the speed of the A-to-D conversion by disabling/enabling a clock prescaler. At low frequencies the leakage currents cause the sample and hold not to work accurately, so IOC2.3 is provided to turn the sample and hold feature off.

A summary of the conversion time for the four options is shown below. The numbers represent the number of state times required for conversion, e.g., 91 states is 22.7 μ s with an 8 MHz XTAL1 (providing a 250 ns state time.)

IOC2.3 1/0 = Sample and Hold off/on

IOC2.4 1/0 = A to D Clock Prescaler off/on

10 MHz XTAL1 maximum with prescaler off

	Clock Prescaler On IOC2.4 = 0	Clock Prescaler Off IOC2.4 = 1	
IOC2.3 = 0 with S&H	158 states 26.33 μ s @ 12 MHz	91 states 22.75 μ s @ 8 MHz	91 states 18.2 μ s @ 10 MHz
IOC2.3 = 1 without S&H	293 states 48.83 μ s @ 12 MHz	163 states 40.75 μ s @ 8 MHz	163 states 32.6 μ s @ 10 MHz

3.7 PULSE-WIDTH-MODULATION OUTPUT (PWM)

The PWM output unit is an 8-bit counter which increments every state time. When the counter equals zero the output is set high, when it equals the value in the PWM register (location 17H) the output goes low. This provides an approximation to an analog output for driving motors and other similar devices. A block diagram of the PWM unit and examples of PWM waveforms are shown in Figures 26 and 27 respectively. The 80C196KA PWM unit has a prescaler bit (divide by 2) which is enabled by setting IOC2.2 = 1. This allows the counter to have a period of 512 state times instead of 256. The PWM frequencies are as follows:

XTAL1 =	8 MHz	10 MHz	12 MHz
IOC2.2 = 0	15.6 KHz	19.6 KHz	23.6 KHz
IOC2.2 = 1	7.8 KHz	9.8 KHz	11.8 KHz

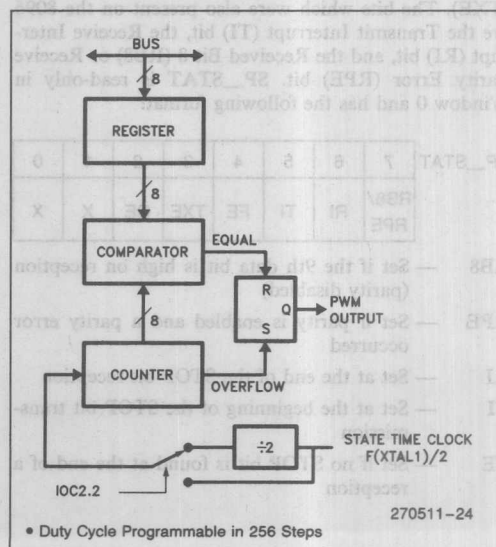


Figure 26. PWM Block Diagram

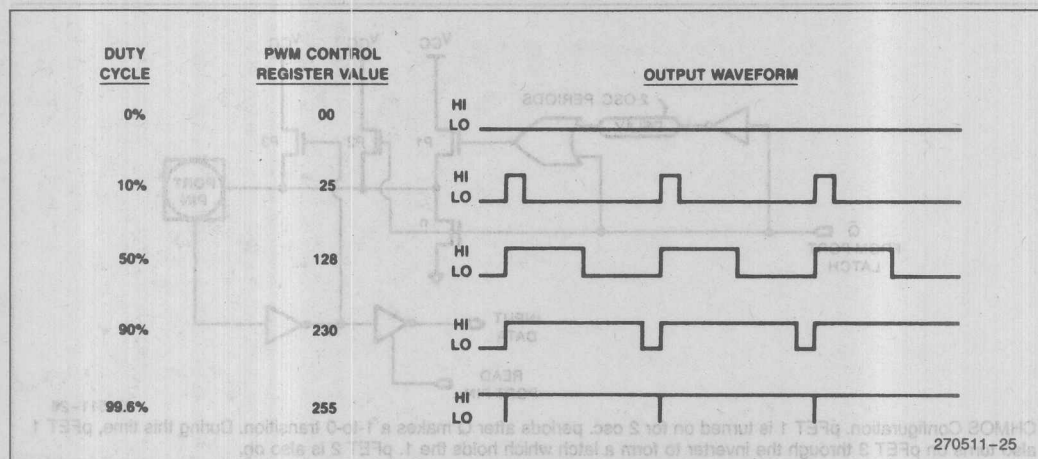


Figure 27. PWM Waveforms

3.8 STANDARD I/O PORTS

Five (5) 8-bit I/O ports are available on the 80C196KA. Port 0 (location 0EH) is an input only port which shares its pins with the A to D converter. Port 1 (location 0FH) is a quasi-bidirectional port. Port 2 (location 10H) has multiple functions on its pins as shown in Figure 28.

Quasi-bidirectional pins can be used as input and output pins without the need for a data direction register. They output a strong low value and a weak high value. The weak high value can be externally pulled low providing an input function. Figure 29 shows the configuration of a CHMOS quasi-bidirectional port. Note that it is not identical to the NMOS version.

Outputting a 0 on a quasi-bidirectional pin turns on the strong pull-down and turns off all of the pull-ups. When a 1 is output the pull-down is turned off and 3 pull-ups (strong-P1, weak-P3, very weak-P2) are turned on. Each time a pin switches from 0 to 1 transistor P1 turns on for two state times. P2 remains on until a zero is written to the pin. P3 is used as a latch, so it is turned on whenever the pin is above the threshold value (around 2V).

To reduce the amount of current which flows when the pin is externally pulled low, P3 is turned off when the pin voltage drops below the threshold. The current required to pull the pin from a high to a low is at its maximum just prior to the pull-up turning off. An external driver can switch these pins easily. The maximum current required occurs at the threshold voltage and is approximately 700 μ A.

Ports 3 and 4 are open drain I/O ports which share their pins with the System Bus. The port 3 and 4 pins will act as port pins if the EA pin is set for internal access and external memory is not being accessed. In all other cases the ports must be reconstructed with external hardware since the system bus uses the pins. Since external memory is always required with the 80C196KA, these ports must be reconstructed by placing latches at addresses 1FFE and 1FFFH in external memory. Future ROM and EPROM parts will be able to use the on-chip ports. By using the port reconstruction feature it is possible to build a multi-chip system which is exactly software compatible with a single-chip system.

PIN	FUNC.	ALTERNATE FUNCTION	CONTROL REG.
2.0	Output	TXD (Serial Port Transmit)	IOC1.5
2.1	Input	RXD (Serial Port Receive)	SPCON.3
2.3	Input	T2CLK (Timer2 Clock & Baud)	IOC0.7
2.4	Input	T2RST (Timer2 Reset)	IOC0.5
2.5	Output	PWM Output	IOC1.0
2.6	QBD*	Timer2 up/down select	IOC2.1
2.7	QBD*	Timer2 Capture	N/A

*QBD = Quasi-bidirectional

Figure 28. Port 2 Multiple Functions

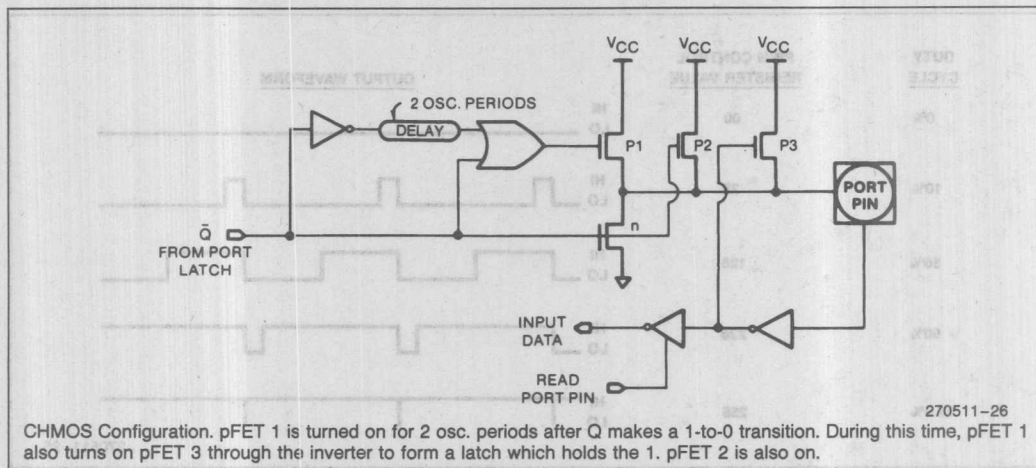


Figure 29. CHMOS Quasi-bidirectional Port Circuit

3.9 USING THE ALTERNATE REGISTER WINDOW (WSR = 15)

I/O register expansion on the new CHMOS members of the MCS-96 family has been provided by making two register windows available. Switching between these windows is done using the Window Select Register (WSR). The PUSHA and POPA instructions can be used to push and pop the WSR and second interrupt mask when entering or leaving interrupts, so it is easy to change between windows.

On the 80C196KA only Window 0 and Window 15 are active. Window 0 is a true superset of the standard MCS-96 SFR space, while Window 15 allows the read-only registers to be written and write-only registers to be read. The only major exception to this is the Timer2 register which is the Timer2 capture register in Window 15. The writeable register for Timer2 is in Window 0. There are also some minor changes and cautions. The descriptions of the registers which have different functions in Window 15 than in Window 0 are listed below:

- AD_COMMAND (02H) — Read the last written command
- AD_RESULT (02H, 03H) — Write a value into the result register
- HSI_MODE (03H) — Read the value in HSI_MODE
- HSI_TIME (04H, 05H) — Write to FIFO Holding register
- HSO_TIME (04H, 05H) — Read the last value placed in the holding register
- HSI_STATUS (06H) — Write to status bits but not to HSI pin bits. (Pin bits are 1,3,5,7).
- HSO_COMMAND (06H) — Read the last value placed in the holding register
- SBUF(RX) (07H) — Write a value into the receive buffer
- SBUF(TX) (07H) — Read the last value written to the transmit buffer
- WATCHDOG(0AH) — Read the value in the upper byte of the WDT
- TIMER1 (0AH, 0BH) — Write a value to Timer1
- TIMER2 (0CH, 0DH) — Read/Write the Timer2 capture register.
(Timer2 read/write is done with WSR = 0)
- IOC2 (0BH) — Last written value is readable, except bit 7 (note 1)
- BAUD_RATE (0EH) — No function, cannot be read
- PORT0 (0EH) — No function, no output drivers on the pins
- SP_STAT (11H) — Set the status bits, TI and RI can be set, but it will not cause an interrupt

- SP_CON (11H) — Read the current control byte
- IOS0 (15H) — Writing to this register controls the HSO pins. Bits 6 and 7 are inactive for writes.
- IOC0 (15H) — Last written value is readable, except bit 1 (note 1)
- IOS1 (16H) — Writing to this register will set the status bits, but not cause interrupts. Bits 6 and 7 are not functional
- IOC1 (16H) — Last written value is readable
- IOS2 (17H) — Writing to this register will set the status bits, but not cause interrupts.
- PWM_CONTROL (17H) — Read the duty cycle value written to PWM_CONTROL

NOTE:

- 1. IOC2.7 (CAM CLEAR) and IOC0.1 (T2RST) are not latched and will read as a 1 (precharged bus).

Being able to write to the read-only registers and vice-versa provides a lot of flexibility. One of the most useful advantages is the ability to set the timers and HSO lines for initial conditions other than zero.

3.10 SFR BIT SUMMARY

A summary of the SFRs which control I/O functions has been included in this section. The summary is separated into a list of those SFRs which have changed on the 80C196KA and a list of those which have remained the same. (The Read and Write comments indicate the register's function in Window 0 unless otherwise specified.)

SBUF(TX) — Now double buffered

07h
write

BAUD RATE — Uses new Baud Rate Values

0Eh
write

SP_STAT:

11h
read

7	6	5	4	3	2	1	0
RB8/ RFE	RI	TI	FE	TXE	OE	X	X

- RPE: Receive Parity Error
- RI: Receive Indicator
- TI: Transmit Indicator
- FE: Framing Error
- TXE: Transmitter Empty
- OE: Receive Overrun Error

**IPEND1:
IMASK1:**

12h,13h
read/write

7	6	5	4	3	2	1	0
NMI	FIFO FULL	EXT INT	T2 OVF	T2 CAP	HSI4	RI	TI

- NMI: Non-Maskable Interrupt
- FIFO FULL: HSIO FIFO full
- EXTINT: External Interrupt Pin
- T2OVF: Timer2 Overflow
- T2CAP: Timer2 Capture
- HSI4: HSI has 4 or more entries in FIFO
- RI: Receive Interrupt
- TI: Transmit Interrupt

7	6	5	4	3	2	1	0
X	X	X	X	W	W	W	W

14h
read/write

WWWW = 0 : SFRs function like a superset of 8096 SFRs
 WWW = 15 : Exchange read/write registers
 WWW = OTHER : Undefined, do not use
 XXXX = 0000B : These bits must always be written as zeros to provide compatibility with future products.

IOS2:

7	6	5	4	3	2	1	0
START A2D	T2 RESET	HSO.5	HSO.4	HSO.3	HSO.2	HSO.1	HSO.0

17h
read

Indicates which HSO event occurred

START A2D : HSO_CMD 15, start A to D

T2RESET : HSO_CMD 14, Timer 2 reset

HSO.0-5 : Output pins HSO.0 through HSO.5

IOC2:

7	6	5	4	3	2	1	0
CLEAR CAM	ENA LOCK	T2ALT INT	A2D CPD	NOSH	SLOW PWM	T2UD ENA	FAST T2EN

0BH
write

CLEAR_CAM : Clear Entire CAM

ENA_LOCK : Enable lockable CAM entry feature

T2ALT INT : Enable T2 Alternate Interrupt at 8000H

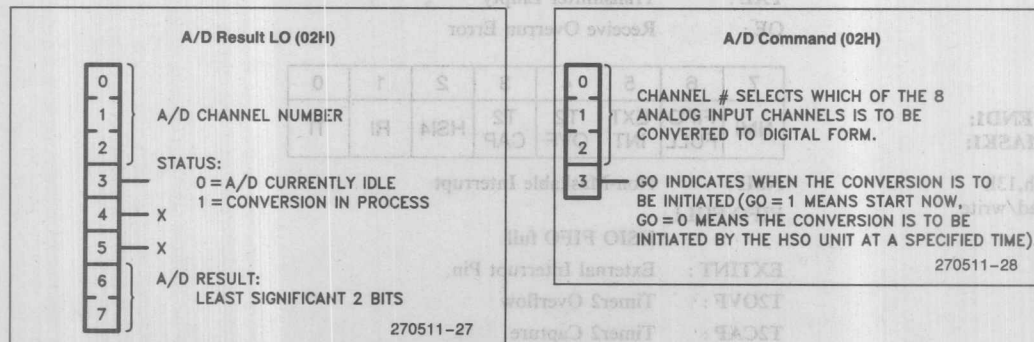
A2D_CPD : Clock Prescale Disable for low XTAL frequency (A to D conversion in fewer state times)

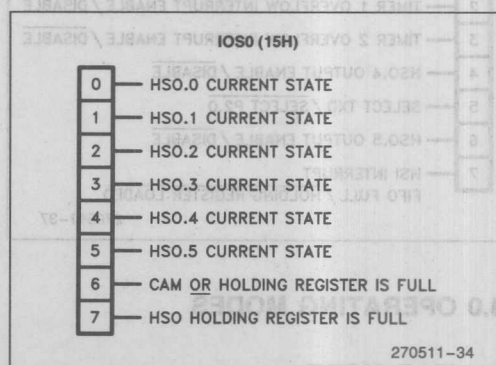
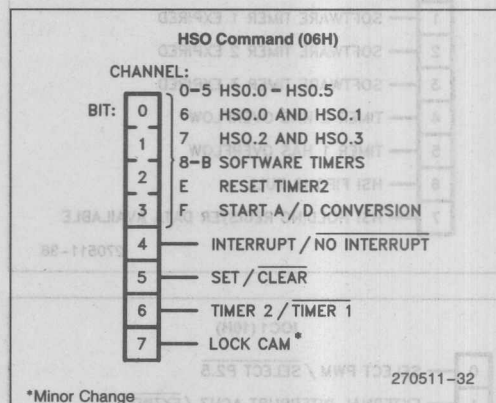
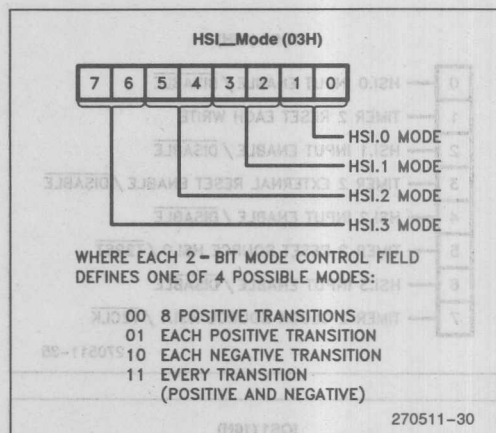
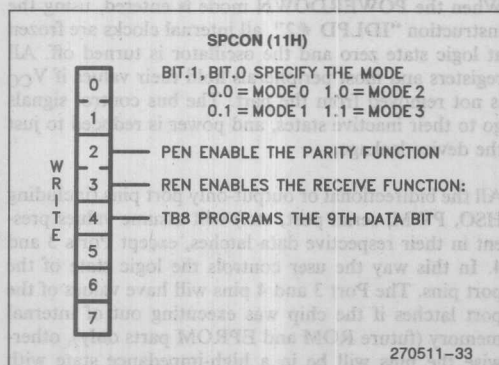
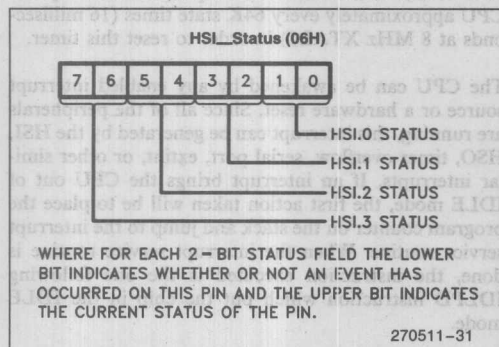
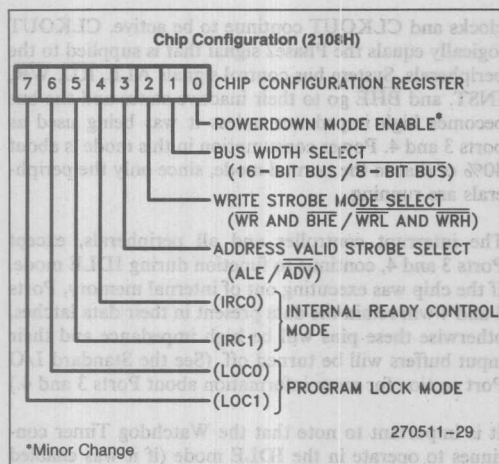
NOSH : Disable A/D Sample and Hold

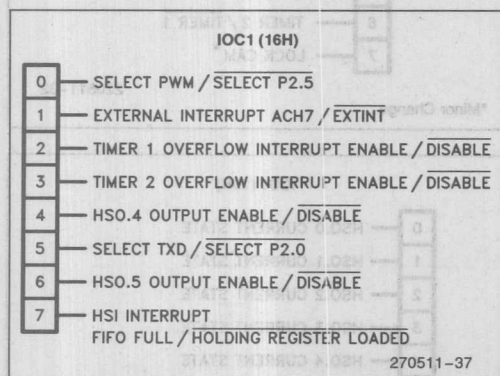
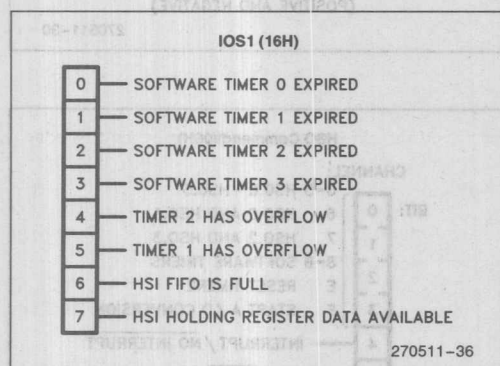
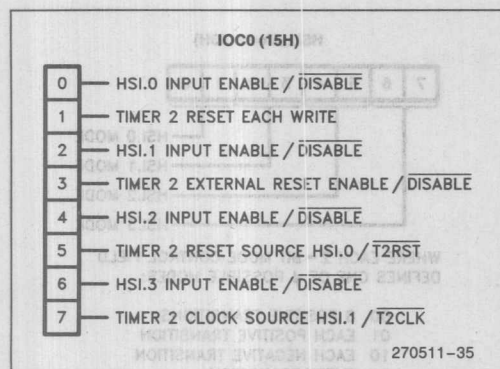
SLOW_PWM : Turn on divide by 2 Prescaler on PWM

T2UD ENA : Enable Timer 2 as up/down counter

FAST_T2EN : Enable Fast increment of T2; once per state time.







4.0 OPERATING MODES

4.1 IDLE MODE

When the IDLE mode is entered, using the instruction "IDLDP #1", the CPU stops executing. The CPU clocks are frozen at logic state zero, but the peripheral

clocks and CLKOUT continue to be active. CLKOUT logically equals the Phase2 signal that is supplied to the peripherals. System bus control signals ALE, RD, WR, INST, and BHE go to their inactive states and the bus becomes high impedance unless it was being used as ports 3 and 4. Power consumption in this mode is about 40% of that in the normal mode, since only the peripherals are running.

The interrupt controller and all peripherals, except Ports 3 and 4, continue to function during IDLE mode. If the chip was executing out of internal memory, Ports 3 and 4 will retain the data present in their data latches, otherwise these pins will be high impedance and their input buffers will be turned off. (See the Standard I/O Port section for more information about Ports 3 and 4.)

It is important to note that the Watchdog Timer continues to operate in the IDLE mode (if it was enabled after reset). This means the chip must wake up the CPU approximately every 64K state times (16 milliseconds at 8 MHz XTAL1) in order to reset this timer.

The CPU can be awakened by any enabled interrupt source or a hardware reset. Since all of the peripherals are running, this interrupt can be generated by the HSI, HSO, timer overflow, serial port, extint, or other similar interrupts. If an interrupt brings the CPU out of IDLE mode, the first action taken will be to place the program counter on the stack and jump to the interrupt service routine. When the interrupt service routine is done, the instruction executed is the one following IDLPD instruction which put the chip in the IDLE mode.

4.2 POWERDOWN MODE

When the POWERDOWN mode is entered, using the instruction "IDLDP #2", all internal clocks are frozen at logic state zero and the oscillator is turned off. All registers and most peripherals hold their values if VCC is not removed from the part. The bus control signals go to their inactive states, and power is reduced to just the device leakage.

All the bidirectional or output-only port pins (including HSO, PWM, serial port, etc.) will assume values present in their respective data latches, except Ports 3 and 4. In this way the user controls the logic state of the port pins. The Port 3 and 4 pins will have values of the port latches if the chip was executing out of internal memory (future ROM and EPROM parts only), otherwise the pins will be in a high-impedance state with input buffers shut off.

All peripherals should be in an inactive state before putting the chip in powerdown. If the A to D converter is in the middle of a conversion it is aborted. The

HSIO, timers (Timer1 and Timer2), and the serial port stop in POWERDOWN mode. If the chip comes out of POWERDOWN by an external interrupt, the serial port will continue from where it left off with a chance of erroneous data transmitted or received. Therefore, the user must shut off the transmitter (not write anything to it) and the receiver ($REN=0$) before putting the chip in POWERDOWN.

When the chip is in Powerdown, it is impossible to time out the Watchdog Timer or detect oscillator failure. Therefore, systems which will use Powerdown should not enable the Watchdog Timer and the systems using the Watchdog Timer should not go into Powerdown, unless the Watchdog is always reset immediately before entering and after exiting Powerdown.

To prevent accidental entry into Powerdown, the Powerdown feature can be disabled at reset by clearing bit 0 of the Chip Configuration Register (CCR). Since the default value of the Configuration Byte is 0FFH, Powerdown is normally enabled.

When in Powerdown, almost the entire state of the 80C196KA will be preserved, not just the most significant 16 bytes of register file. The V_{CC} (not V_{PP}) is used to supply power to the chip, so it must remain within specifications if the chip status is to be maintained. Certain SFRs, may contain incorrect information when the chip comes out of Powerdown. SFRs which could do this are the A/D result and serial port registers since the functions of these registers are real-time dependent and CPU-time stops in Powerdown mode. A/D commands in progress are aborted when coming out of Powerdown. It is the users responsibility to handle the serial port.

The Powerdown mode can be exited using either RESET or an external interrupt pin. If the RESET pin is used, it must externally be held low long enough for the oscillator to stabilize, plus 4 states for the reset sequence.

When exiting Powerdown using an external interrupt, a positive level on the pin mapped to INT7 (either EXTINT pin or Port0.7 pin) will bring the part out of Powerdown mode. This procedure is not affected by either the interrupt disable bit or the interrupt mask register. An internal timing circuit is used to ensure that the oscillator has stabilized before the internal clocks are turned on. Figure 30 shows the power down and powerup sequence in such a case.

During normal operation, before the chip goes into powerdown, the V_{PP} pin will rise to V_{CC} through an internal pullup. The user must connect a capacitor between V_{PP} and V_{SS} . A positive level on the pin mapped to INT7 (external interrupt) will start discharging this capacitor if the chip was in Powerdown when this edge occurred. The internal current source used to discharge this capacitor is approximately 100 μA . A threshold detector will detect 1 V or lower on the V_{PP} pin and mark the end of the time-out period. A 1 μF capacitor will provide about 4 ms startup time.

If the external interrupt is used to bring the part out of Powerdown, that bit will be set in the interrupt pending register when the chip starts to run. If the interrupt is not masked off, the first section of code executed will be the interrupt service routine, otherwise execution will begin with the code following the IDLPD instruction. If the interrupt is not serviced the interrupt pending bit will remain set.

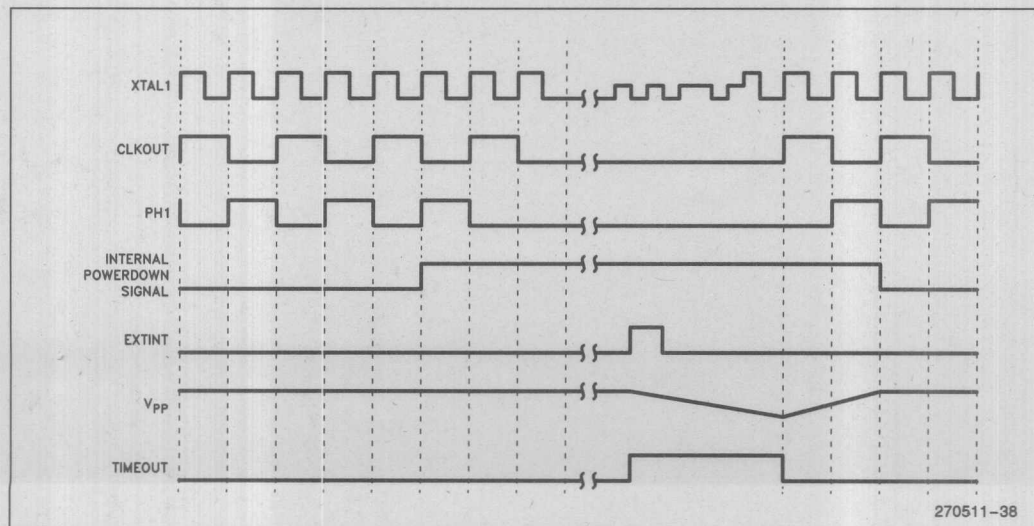


Figure 30. Powerdown/Up Sequence

4.3 RESET SEQUENCE AND STATUS

Figure 31 shows the sequence used on the 80C196KA.

As soon as the RESET line is pulled low the I/O and control lines will go into their reset condition. The state of these lines is shown below:

Pin Name	Multiplexed Port Pins	Value of the Pin on Reset
RESET		Mid-sized Pullup
ALE		Weak Pullup
RD		Weak Pullup
BHE		Weak Pullup
WR		Weak Pullup
INST		Weak Pull-up
EA		Undefined Input *
READY		Undefined Input *
NMI		Undefined Input *
BUSWIDTH		Undefined Input *
CLKOUT		Phase 2 of Clock
System Bus	P3.0-P4.7	Weak Pullups

The weak pullups and pulldowns are sufficient to hold a line in one position or another. Pins listed as undefined inputs (*) must be tied or driven externally, otherwise the part may not function properly. Reset must be held low for 4 state times.

In order for the part to function, the following pins must be connected:

V_{CC}, V_{SS1}, V_{SS2}, V_{REF}, ANGND, XTAL1, XTAL2

Pin Name	Multiplexed Port Pins	Value of the Pin on Reset
ACH0-7	P0.0-P0.7	Undefined Input *
PORT1	P1.0-P1.7	Weak Pullups
TXD	P2.0	Weak Pullup
RXD	P2.1	Undefined Input *
EXTINT	P2.2	Undefined Input *
T2CLK	P2.3	Undefined Input *
T2RST	P2.4	Undefined Input *
PWM	P2.5	Weak Pulldown
—	P2.6-P2.7	Weak Pullups
HSI0-HSI1		Undefined Input *
HSI2/HSO4		Undefined Input *
HSI3/HSO5		Undefined Input *
HSO0-HSO3		Weak Pulldown

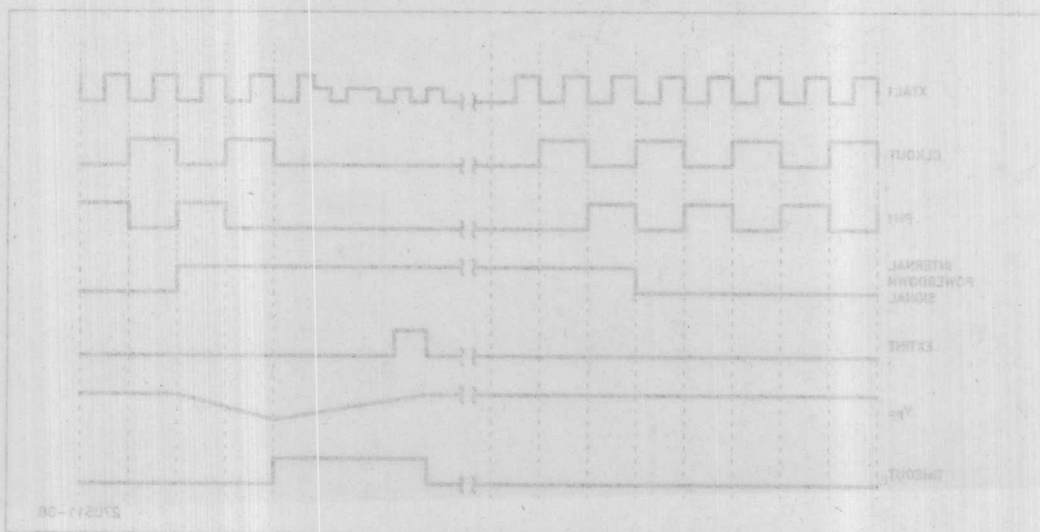


Figure 30. Powerdown/Up Sequence

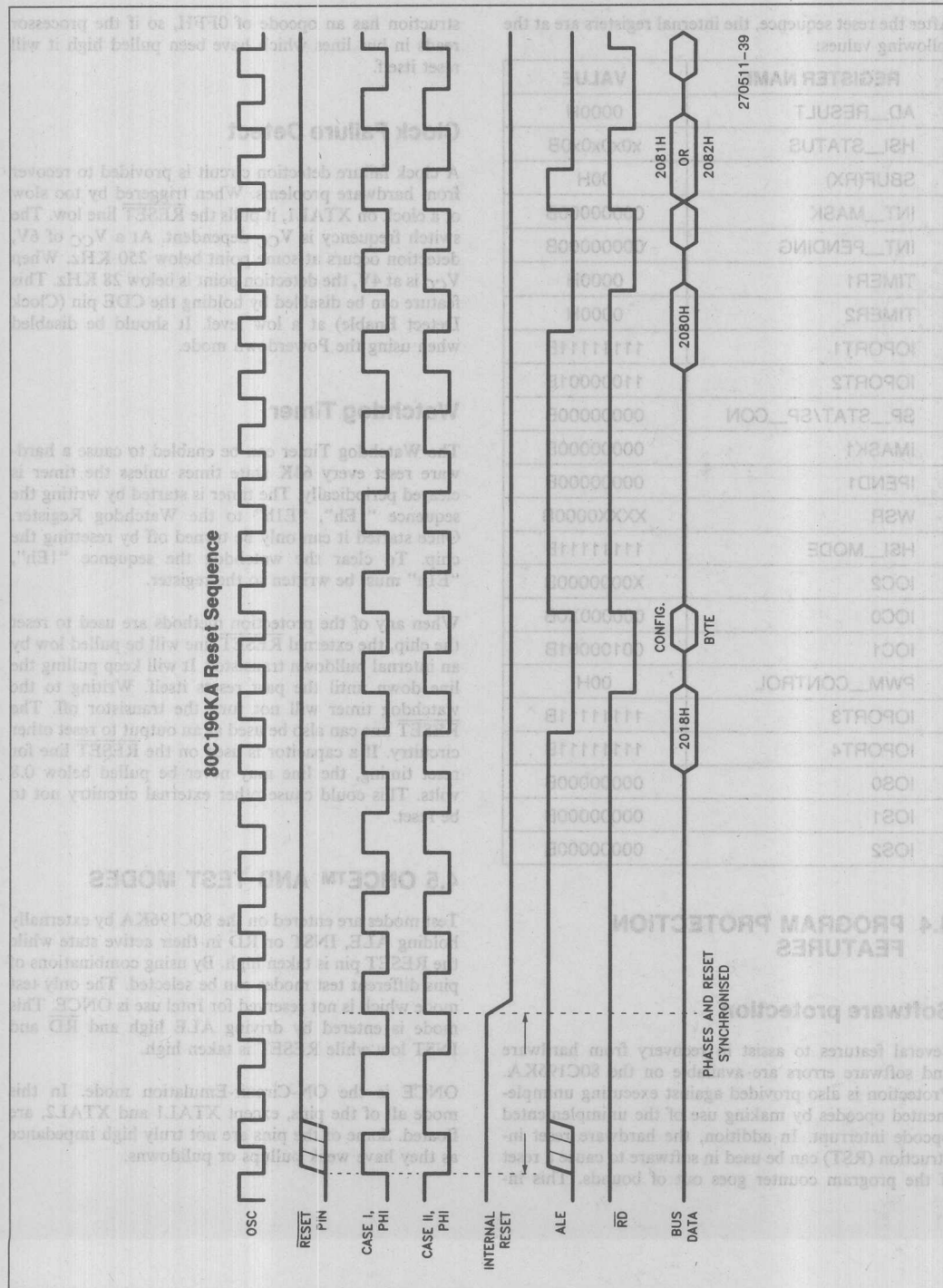


Figure 31. Reset Sequence

After the reset sequence, the internal registers are at the following values:

REGISTER NAME	VALUE
AD_RESULT	0000H
HSI_STATUS	x0x0x0x0B
SBUF(RX)	00H
INT_MASK	00000000B
INT_PENDING	00000000B
TIMER1	0000H
TIMER2	0000H
IOPORT1	11111111B
IOPORT2	11000001B
SP_STAT/SP_CON	00000000B
IMASK1	00000000B
IPEND1	00000000B
WSR	XXXX0000B
HSI_MODE	11111111B
IOC2	X0000000B
IOC0	000000X0B
IOC1	00100001B
PWM_CONTROL	00H
IOPORT3	11111111B
IOPORT4	11111111B
IOS0	00000000B
IOS1	00000000B
IOS2	00000000B

4.4 PROGRAM PROTECTION FEATURES

Software protection

Several features to assist in recovery from hardware and software errors are available on the 80C196KA. Protection is also provided against executing unimplemented opcodes by making use of the unimplemented opcode interrupt. In addition, the hardware reset instruction (RST) can be used in software to cause a reset if the program counter goes out of bounds. This in-

struction has an opcode of OFFH, so if the processor reads in bus lines which have been pulled high it will reset itself.

Clock Failure Detect

A clock failure detection circuit is provided to recover from hardware problems. When triggered by too slow of a clock on XTAL1, it pulls the RESET line low. The switch frequency is V_{CC} dependent. At a V_{CC} of 6V, detection occurs at some point below 250 KHz. When V_{CC} is at 4V, the detection point is below 28 KHz. This feature can be disabled by holding the CDE pin (Clock Detect Enable) at a low level. It should be disabled when using the Powerdown mode.

Watchdog Timer

The Watchdog Timer can be enabled to cause a hardware reset every 64K state times unless the timer is cleared periodically. The timer is started by writing the sequence "1Eh", "E1h" to the Watchdog Register. Once started it can only be turned off by resetting the chip. To clear the watchdog the sequence "1Eh", "E1h" must be written to the register.

When any of the protection methods are used to reset the chip, the external RESET line will be pulled low by an internal pulldown transistor. It will keep pulling the line down until the part resets itself. Writing to the watchdog timer will not turn the transistor off. The RESET line can also be used as an output to reset other circuitry. If a capacitor is used on the RESET line for reset timing, the line may never be pulled below 0.8 volts. This could cause other external circuitry not to be reset.

4.5 ONCETM AND TEST MODES

Test modes are entered on the 80C196KA by externally holding ALE, INST or RD in their active state while the RESET pin is taken high. By using combinations of pins different test modes can be selected. The only test mode which is not reserved for Intel use is ONCE. This mode is entered by driving ALE high and RD and INST low while RESET is taken high.

ONCE is the ON-Circuit-Emulation mode. In this mode all of the pins, except XTAL1 and XTAL2, are floated. Some of the pins are not truly high impedance as they have weak pullups or pulldowns.

Figure 31. Reset Sequence

5.0 CONVERTING FROM OTHER MCS®-96 PRODUCTS TO THE 80C196KA

The following list of suggestions for designing an 8X9XBH system will yield a design that is easily converted to the 80C196KA.

1. Do not base critical timing loops on instruction or peripheral execution times.
2. Use equate statements to set all timing parameters, including the baud rate.
3. Do not base hardware timings on CLKOUT or XTAL1. The timings of the 80C196KA are different

than those of the 8X9XBH, but they will function with standard ROM / EPROM / Peripheral type memory systems.

4. Make sure all inputs are tied high or low and not left floating.
5. The BHE/WRH signal is not valid in the 8-bit mode.
6. Indexed and Indirect operations relative to the stack pointer (SP) work differently on the 80C196KA than on the 8096BH. On the 8096BH, the address is calculated based on the un-updated version of the SP. The 80C196KA uses the updated version. The offset for PUSH[SP], POP[SP], PUSHnn[SP] and POPnn[SP] instructions may need to be changed by a count of 2.

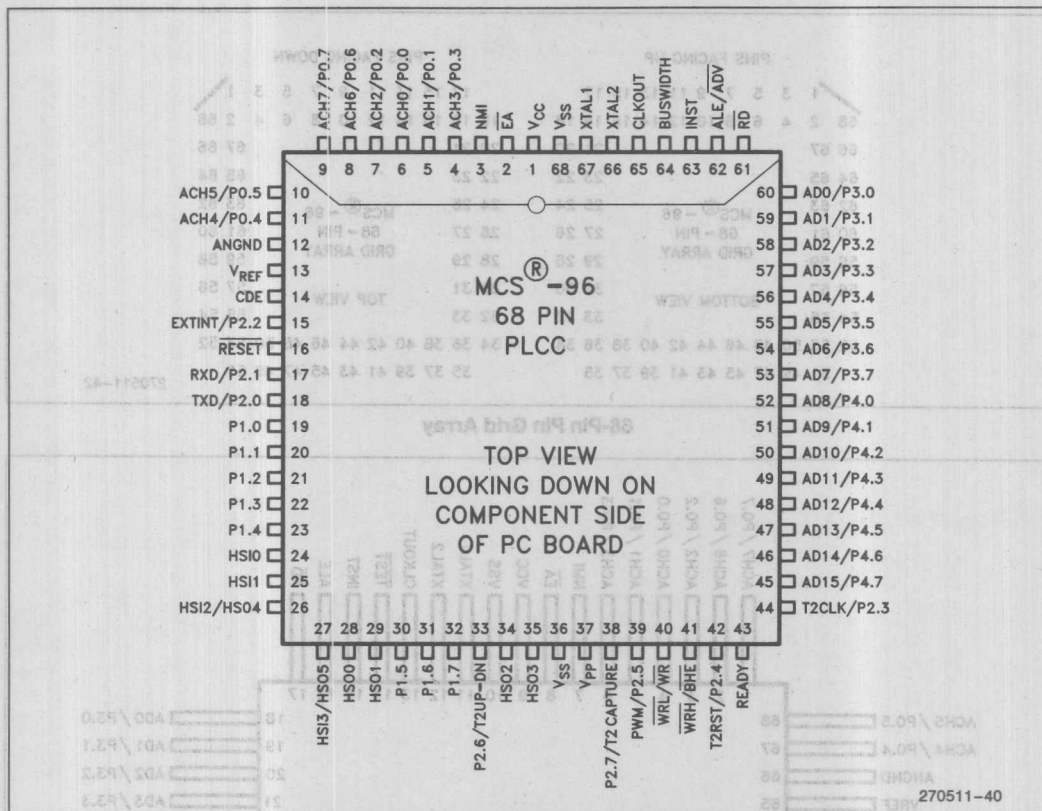
AD81P4.0	50	50	AD81P4.0	50	50	AD81P4.0	50	50
AD81P4.1	51	51	AD81P4.1	51	51	AD81P4.1	51	51
AD10P4.2	52	52	AD10P4.2	52	52	AD10P4.2	52	52
AD11P4.3	53	53	AD11P4.3	53	53	AD11P4.3	53	53
AD12P4.4	54	54	AD12P4.4	54	54	AD12P4.4	54	54
AD13P4.5	55	55	AD13P4.5	55	55	AD13P4.5	55	55
AD14P4.6	56	56	AD14P4.6	56	56	AD14P4.6	56	56
AD15P4.7	57	57	AD15P4.7	57	57	AD15P4.7	57	57
TCLKP5.3	58	58	TCLKP5.3	58	58	TCLKP5.3	58	58
READY	59	59	READY	59	59	READY	59	59
TSTRP5.4	60	60	TSTRP5.4	60	60	TSTRP5.4	60	60
BRE/WRH	61	61	BRE/WRH	61	61	BRE/WRH	61	61
WR/WRH	62	62	WR/WRH	62	62	WR/WRH	62	62
PWMF5.5	63	63	PWMF5.5	63	63	PWMF5.5	63	63
P5.7VTSAPTURE	64	64	P5.7VTSAPTURE	64	64	P5.7VTSAPTURE	64	64
Vpp	65	65	Vpp	65	65	Vpp	65	65
Vss	66	66	Vss	66	66	Vss	66	66
H80.3	67	67	H80.3	67	67	H80.3	67	67
H80.2	68	68	H80.2	68	68	H80.2	68	68
P5.8VTSUP-DN	69	69	P5.8VTSUP-DN	69	69	P5.8VTSUP-DN	69	69
P1.7	70	70	P1.7	70	70	P1.7	70	70
AD81P4.0	71	71	AD81P4.0	71	71	AD81P4.0	71	71
AD81P4.1	72	72	AD81P4.1	72	72	AD81P4.1	72	72
AD10P4.2	73	73	AD10P4.2	73	73	AD10P4.2	73	73
AD11P4.3	74	74	AD11P4.3	74	74	AD11P4.3	74	74
AD12P4.4	75	75	AD12P4.4	75	75	AD12P4.4	75	75
AD13P4.5	76	76	AD13P4.5	76	76	AD13P4.5	76	76
AD14P4.6	77	77	AD14P4.6	77	77	AD14P4.6	77	77
AD15P4.7	78	78	AD15P4.7	78	78	AD15P4.7	78	78
TCLKP5.3	79	79	TCLKP5.3	79	79	TCLKP5.3	79	79
READY	80	80	READY	80	80	READY	80	80
TSTRP5.4	81	81	TSTRP5.4	81	81	TSTRP5.4	81	81
BRE/WRH	82	82	BRE/WRH	82	82	BRE/WRH	82	82
WR/WRH	83	83	WR/WRH	83	83	WR/WRH	83	83
PWMF5.5	84	84	PWMF5.5	84	84	PWMF5.5	84	84
P5.7VTSAPTURE	85	85	P5.7VTSAPTURE	85	85	P5.7VTSAPTURE	85	85
Vpp	86	86	Vpp	86	86	Vpp	86	86
Vss	87	87	Vss	87	87	Vss	87	87
H80.3	88	88	H80.3	88	88	H80.3	88	88
H80.2	89	89	H80.2	89	89	H80.2	89	89
P5.8VTSUP-DN	90	90	P5.8VTSUP-DN	90	90	P5.8VTSUP-DN	90	90
P1.7	91	91	P1.7	91	91	P1.7	91	91
AD81P4.0	92	92	AD81P4.0	92	92	AD81P4.0	92	92
AD81P4.1	93	93	AD81P4.1	93	93	AD81P4.1	93	93
AD10P4.2	94	94	AD10P4.2	94	94	AD10P4.2	94	94
AD11P4.3	95	95	AD11P4.3	95	95	AD11P4.3	95	95
AD12P4.4	96	96	AD12P4.4	96	96	AD12P4.4	96	96
AD13P4.5	97	97	AD13P4.5	97	97	AD13P4.5	97	97
AD14P4.6	98	98	AD14P4.6	98	98	AD14P4.6	98	98
AD15P4.7	99	99	AD15P4.7	99	99	AD15P4.7	99	99
TCLKP5.3	100	100	TCLKP5.3	100	100	TCLKP5.3	100	100
READY	101	101	READY	101	101	READY	101	101
TSTRP5.4	102	102	TSTRP5.4	102	102	TSTRP5.4	102	102
BRE/WRH	103	103	BRE/WRH	103	103	BRE/WRH	103	103
WR/WRH	104	104	WR/WRH	104	104	WR/WRH	104	104
PWMF5.5	105	105	PWMF5.5	105	105	PWMF5.5	105	105
P5.7VTSAPTURE	106	106	P5.7VTSAPTURE	106	106	P5.7VTSAPTURE	106	106
Vpp	107	107	Vpp	107	107	Vpp	107	107
Vss	108	108	Vss	108	108	Vss	108	108
H80.3	109	109	H80.3	109	109	H80.3	109	109
H80.2	110	110	H80.2	110	110	H80.2	110	110
P5.8VTSUP-DN	111	111	P5.8VTSUP-DN	111	111	P5.8VTSUP-DN	111	111
P1.7	112	112	P1.7	112	112	P1.7	112	112
AD81P4.0	113	113	AD81P4.0	113	113	AD81P4.0	113	113
AD81P4.1	114	114	AD81P4.1	114	114	AD81P4.1	114	114
AD10P4.2	115	115	AD10P4.2	115	115	AD10P4.2	115	115
AD11P4.3	116	116	AD11P4.3	116	116	AD11P4.3	116	116
AD12P4.4	117	117	AD12P4.4	117	117	AD12P4.4	117	117
AD13P4.5	118	118	AD13P4.5	118	118	AD13P4.5	118	118
AD14P4.6	119	119	AD14P4.6	119	119	AD14P4.6	119	119
AD15P4.7	120	120	AD15P4.7	120	120	AD15P4.7	120	120
TCLKP5.3	121	121	TCLKP5.3	121	121	TCLKP5.3	121	121
READY	122	122	READY	122	122	READY	122	122
TSTRP5.4	123	123	TSTRP5.4	123	123	TSTRP5.4	123	123
BRE/WRH	124	124	BRE/WRH	124	124	BRE/WRH	124	124
WR/WRH	125	125	WR/WRH	125	125	WR/WRH	125	125
PWMF5.5	126	126	PWMF5.5	126	126	PWMF5.5	126	126
P5.7VTSAPTURE	127	127	P5.7VTSAPTURE	127	127	P5.7VTSAPTURE	127	127
Vpp	128	128	Vpp	128	128	Vpp	128	128
Vss	129	129	Vss	129	129	Vss	129	129
H80.3	130	130	H80.3	130	130	H80.3	130	130
H80.2	131	131	H80.2	131	131	H80.2	131	131
P5.8VTSUP-DN	132	132	P5.8VTSUP-DN	132	132	P5.8VTSUP-DN	132	132
P1.7	133	133	P1.7	133	133	P1.7	133	133
AD81P4.0	134	134	AD81P4.0	134	134	AD81P4.0	134	134
AD81P4.1	135	135	AD81P4.1	135	135	AD81P4.1	135	135
AD10P4.2	136	136	AD10P4.2	136	136	AD10P4.2	136	136
AD11P4.3	137	137	AD11P4.3	137	137	AD11P4.3	137	137
AD12P4.4	138	138	AD12P4.4	138	138	AD12P4.4	138	138
AD13P4.5	139	139	AD13P4.5	139	139	AD13P4.5	139	139
AD14P4.6	140	140	AD14P4.6	140	140	AD14P4.6	140	140
AD15P4.7	141	141	AD15P4.7	141	141	AD15P4.7	141	141
TCLKP5.3	142	142	TCLKP5.3	142	142	TCLKP5.3	142	142
READY	143	143	READY	143	143	READY	143	143
TSTRP5.4	144	144	TSTRP5.4	144	144	TSTRP5.4	144	144
BRE/WRH	145	145	BRE/WRH	145	145	BRE/WRH	145	145
WR/WRH	146	146	WR/WRH	146	146	WR/WRH	146	146
PWMF5.5	147	147	PWMF5.5	147	147	PWMF5.5	147	147
P5.7VTSAPTURE	148	148	P5.7VTSAPTURE	148	148	P5.7VTSAPTURE	148	148
Vpp	149	149	Vpp	149	149	Vpp	149	149
Vss	150	150	Vss	150	150	Vss	150	150
H80.3	151	151	H80.3	151	151	H80.3	151	151
H80.2	152	152	H80.2	152	152	H80.2	152	152
P5.8VTSUP-DN	153	153	P5.8VTSUP-DN	153	153	P5.8VTSUP-DN	153	153
P1.7	154	154	P1.7	154	154	P1.7	154	154
AD81P4.0	155	155	AD81P4.0	155	155	AD81P4.0	155	155
AD81P4.1	156	156	AD81P4.1	156	156	AD81P4.1	156	156
AD10P4.2	157	157	AD10P4.2	157	157	AD10P4.2	157	157
AD11P4.3	158	158	AD11P4.3	158	158	AD11P4.3	158	158
AD12P4.4	159	159	AD12P4.4	159	159	AD12P4.4	159	159
AD13P4.5	160	160	AD13P4.5	160	160	AD13P4.5	160	160
AD14P4.6	161	161	AD14P4.6	161	161	AD14P4.6	161	161
AD15P4.7	162	162	AD15P4.7	162	162	AD15P4.7	162	162
TCLKP5.3	163	163	TCLKP5.3	163	163	TCLKP5.3	163	163
READY	164	164	READY	164	164	READY	164	164
TSTRP5.4	165	165	TSTRP5.4	165	165	TSTRP5.4	165	165
BRE/WRH	166	166	BRE/WRH	166	166	BRE/WRH	166	166
WR/WRH	167	167	WR/WRH	167	167	WR/WRH	167	167
PWMF5.5	168	168	PWMF5.5	168	168	PWMF5.5	168	168
P5.7VTSAPTURE	169	169	P5.7VTSAPTURE	169	169	P5.7VTSAPTURE	169	169
Vpp	170	170	Vpp	170	170	Vpp	170	170
Vss	171	171	Vss	171	171	Vss	171	171
H80.3	172	172	H80.3	172	172	H80.3	172	172
H80.2	173	173	H80.2	173	173	H80.2	173	173
P5.8VTSUP-DN	174	174	P5.8VTSUP-DN	174	174	P5.8VTSUP-DN	174	174
P1.7	175	175	P1.7	175	175	P1.7	175	175
AD81P4.0	176	176	AD81P4.0	176	176	AD81P4.0	176	176
AD81P4.1	177	177	AD81P4.1	177	177	AD81P4.1	177	177
AD10P4.2	178	178	AD10P4.2	178	178	AD10P4.2	178	178
AD11P4.3	179	179	AD11P4.3	179	179	AD11P4.3	179	179
AD12P4.4	180	180	AD12P4.4	180	180	AD12P4.4	180	180
AD13P4.5	181	181	AD13P4.5	181	181	AD13P4.5	181	181
AD14P4.6	182	182	AD14P4.6	182	182	AD14P4.6	182	182
AD15P4.7	183	183	AD15P4.7	183	183	AD15P4.7	183	183
TCLKP5.3	184	184	TCLKP5.3	184	184	TCLKP5.3	184	184
READY	185	185	READY	185	185	READY	185	185
TSTRP5.4	186	186	TSTRP5.4	186	186	TSTRP5.4	186	186
BRE/WRH	187	187	BRE/WRH	187	187	BRE/WRH	187	187
WR/WRH	188	188	WR/WRH	188	188	WR/WRH	188	188
PWMF5.5	189	189	PWMF5.5	189	189	PWMF5.5	189	189
P5.7VTSAPTURE	190	190	P5.7VTSAPTURE	190	190	P5.7VTSAPTURE	190	190
Vpp	191	191	Vpp	191	191	Vpp	191	19

6.0 PACKAGES, PINOUTS, PIN DEFINITIONS

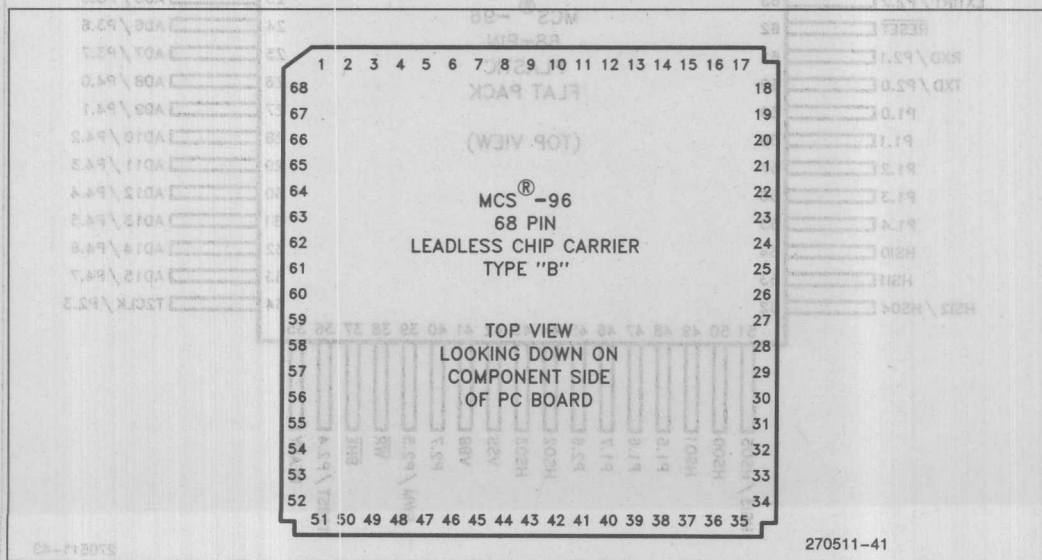
PACKAGING

The 80C196KA is available in 68-pin PLCC and LCC packages. Contact your local sales office to determine the exact ordering code for the part desired.

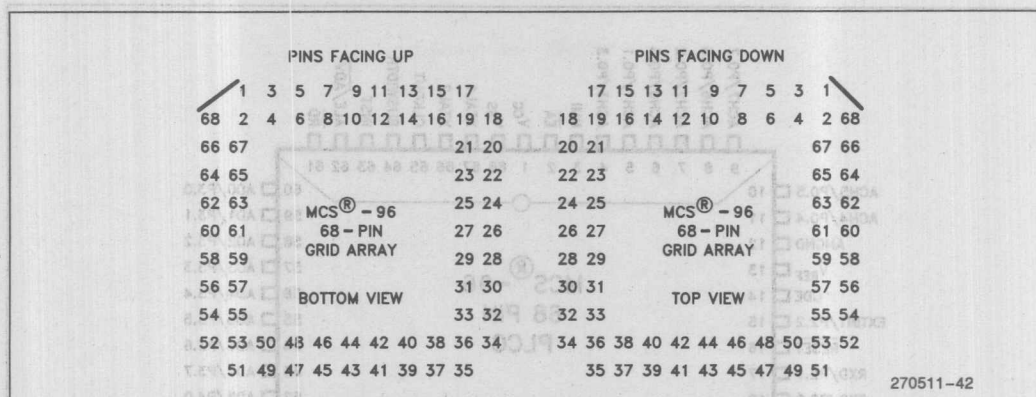
PFP LCC CGA	PLCC	Description	PFP LCC CGA	PLCC	Description	PFP LCC CGA	PLCC	Description
1	9	ACH7/P0.7	24	54	AD6/P3.6	47	31	P1.6
2	8	ACH6/P0.6	25	53	AD7/P3.7	48	30	P1.5
3	7	ACH2/P0.2	26	52	AD8/P4.0	49	29	HSO.1
4	6	ACH0/P0.0	27	51	AD9/P4.1	50	28	HSO.0
5	5	ACH1/P0.1	28	50	AD10/P4.2	51	27	HSO.5/HSI.3
6	4	ACH3/P0.3	29	49	AD11/P4.3	52	26	HSO.4/HSI.2
7	3	NMI	30	48	AD12/P4.4	53	25	HSI.1
8	2	EA	31	47	AD13/P4.5	54	24	HSI.0
9	1	VCC	32	46	AD14/P4.6	55	23	P1.4
10	68	VSS	33	45	AD15/P4.7	56	22	P1.3
11	67	XTAL1	34	44	T2CLK/P2.3	57	21	P1.2
12	66	XTAL2	35	43	READY	58	20	P1.1
13	65	CLKOUT	36	42	T2RST/P2.4	59	19	P1.0
14	64	BUSWIDTH	37	41	BHE/WRH	60	18	TXD/P2.0
15	63	INST	38	40	WR/WRL	61	17	RXD/P2.1
16	62	ALE/ADV	39	39	PWM/P2.5	62	16	RESET
17	61	RD	40	38	P2.7/T2CAPTURE	63	15	EXTINT/P2.2
18	60	AD0/P3.0	41	37	Vpp	64	14	CDE
19	59	AD1/P3.1	42	36	VSS	65	13	VREF
20	58	AD2/P3.2	43	35	HSO.3	66	12	ANGND
21	57	AD3/P3.3	44	34	HSO.2	67	11	ACH4/P0.4
22	56	AD4/P3.4	45	33	P2.6/T2UP-DN	68	10	ACH5/P0.5
23	55	AD5/P3.5	46	32	P1.7			



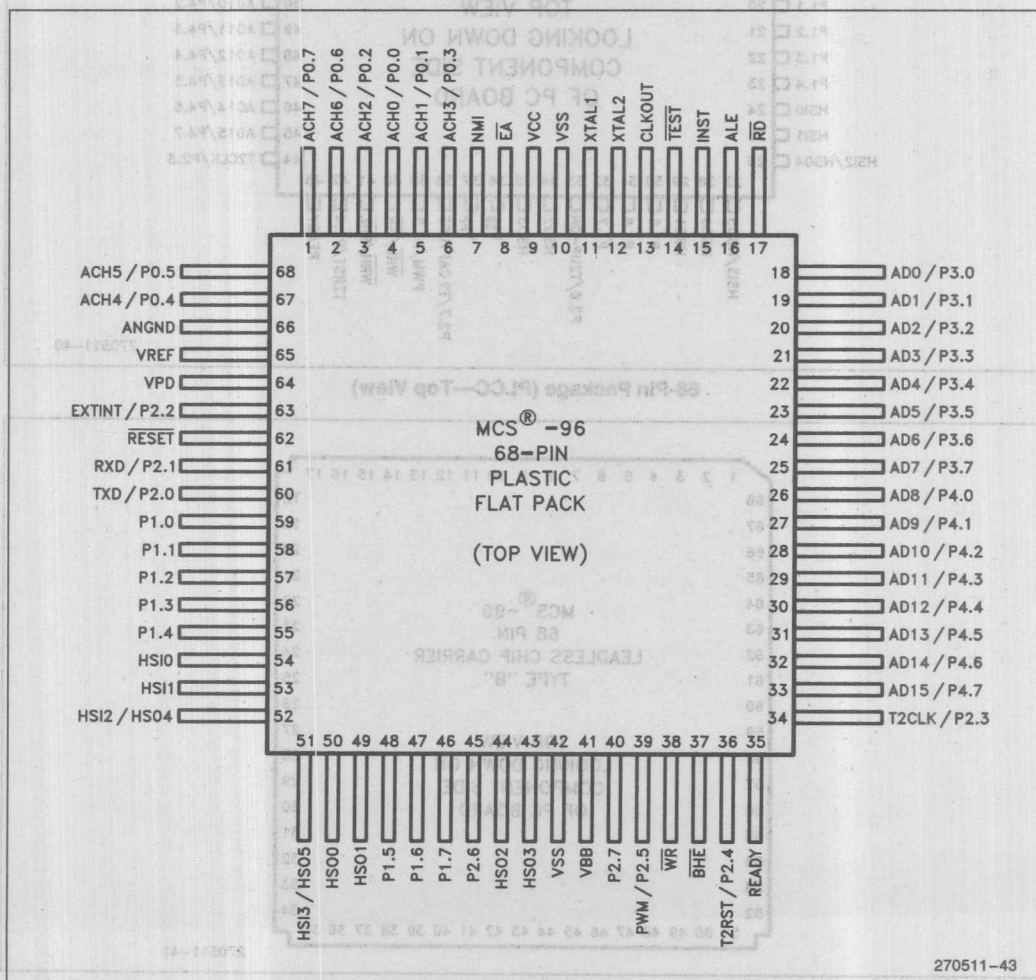
68-Pin Package (PLCC—Top View)



68-Pin Package (LCC—Top View)



68-Pin Pin Grid Array



68-Pin Flatpack

PIN DESCRIPTIONS

Symbol	Name and Function
V _{CC}	Main supply voltage (5V).
V _{SS}	Digital circuit ground (0V). There are two V _{SS} pins, both of which must be connected.
CDE	Clock Detect Enable - When pulled high enables the clock failure detection circuit. If the XTAL1 frequency falls below a specified limit the RESET pin will be pulled low.
V _{REF}	Reference voltage for the A/D converter (5V). V _{REF} is also the supply voltage to the analog portion of the A/D converter and the logic used to read Port 0. Must be connected for A/D and Port 0 to function.
ANGND	Reference ground for the A/D converter. Must be held at nominally the same potential as V _{SS} .
V _{PP}	Timing pin for the return from powerdown circuit. Connect this pin with a 1 μ F capacitor to V _{SS} and a 1 M Ω resistor to V _{CC} . If this function is not used V _{PP} may be tied to V _{CC} .
XTAL1	Input of the oscillator inverter and of the internal clock generator.
XTAL2	Output of the oscillator inverter.
CLKOUT	Output of the internal clock generator. The frequency of CLKOUT is 1/2 the oscillator frequency. It has a 50% duty cycle.
RESET	Reset input to the chip. Input low for at least 4 state times to reset the chip. The subsequent low-to-high transition re-synchronizes CLKOUT and commences a 10-state-time sequence in which the PSW is cleared, a byte read from 2018H loads CCR, and a jump to locations 2080H is executed. Input high for normal operation. RESET has an internal pullup.
BUSWIDTH	Input for buswidth selection. If CCR bit 1 is a one, this pin selects the bus width for the bus cycle in progress. If BUSWIDTH is a 1, a 16-bit bus cycle occurs. If BUSWIDTH is a 0 an 8-bit cycle occurs. If CCR bit 1 is a 0, the bus is always an 8-bit bus.
NMI	A positive transition causes a vector through 203EH.
INST	Output high during an external memory read indicates the read is an instruction fetch. INST is valid throughout the bus cycle. INST is activated only during external memory accesses.
EA	Input for memory select (External Access). EA equal to a TTL-high causes memory accesses to locations 2000H through 3FFFH to be directed to on-chip ROM/EPROM. EA equal to a TTL-low causes accesses to these locations to be directed to off-chip memory.
ALE/ADV	Address Latch Enable or Address Valid output, as selected by CCR. Both pin options provide a latch to demultiplex the address from the address/data bus. When the pin is ADV, it goes inactive high at the end of the bus cycle. ADV can be used as a chip select for external memory. ALE/ADV is activated only during external memory accesses.
RD	Read signal output to external memory. RD is activated only during external memory reads.
WR/WRL	Write and Write Low output to external memory, as selected by the CCR. WR will go low for every external write, while WRL will go low only for external writes where an even byte is being written. WR/WRL is activated only during external memory writes.
BHE/WRH	Bus High Enable or Write High output to external memory, as selected by the CCR. BHE = 0 selects the bank of memory that is connected to the high byte of the data bus. A0 = 0 selects the bank of memory that is connected to the low byte of the data bus. Thus accesses to a 16-bit wide memory can be to the low byte only (A0 = 0, BHE = 1), to the high byte only (A0 = 1, BHE = 0), or both bytes (A0 = 0, BHE = 0). If the WRH function is selected, the pin will go low if the bus cycle is writing to an odd memory location. BHE/WRH is valid only during 16-bit external memory write cycles.

PIN DESCRIPTIONS (Continued)

Symbol	Name and Function
READY	Ready input to lengthen external memory cycles, for interfacing to slow or dynamic memory, or for bus sharing. If the pin is high, CPU operation continues in a normal manner. If the pin is low prior to the falling edge of CLKOUT, the memory controller goes into a wait mode until the next positive transition in CLKOUT occurs with READY high. When the external memory is not being used, READY has no effect. Internal control of the number of wait states inserted into a bus cycle held not ready is available through configuration of CCR.
HSI	Inputs to High Speed Input Unit. Four HSI pins are available: HSI.0, HSI.1, HSI.2, and HSI.3. Two of them (HSI.2 and HSI.3) are shared with the HSO Unit. The HSI pins are also used as inputs by future EPROM parts in Programming Mode.
HSO	Outputs from High Speed Output Unit. Six HSO pins are available: HSO.0, HSO.1, HSO.2, HSO.3, HSO.4, and HSO.5. Two of them (HSO.4 and HSO.5) are shared with the HSI Unit.
Port 0	8-bit high impedance input-only port. These pins can be used as digital inputs and/or as analog inputs to the on-chip A/D converter. These pins are also a mode input to future EPROM parts in the Programming Mode.
Port 1	8-bit quasi-bidirectional I/O port.
Port 2	8-bit multi-functional port. All of its pins are shared with other functions in the 80C196KA.
Ports 3 and 4	8-bit bi-directional I/O ports with open drain outputs. These pins are shared with the multiplexed address/data bus which has strong internal pullups. Available only on future ROM and EPROM parts.
BUSWIDTH	Input for buswidth selection. If CCR bit 1 is a one, this pin selects the bus width for the bus cycle in progress. If BUSWIDTH is a 1, a 16-bit bus cycle occurs. If BUSWIDTH is a 0 an 8-bit cycle occurs. If CCR bit 1 is a 0, the bus is always an 8-bit bus.
INT	A positive transition causes a vector through S03EH.
INST	Output high during an external memory read indicates the read is an instruction fetch. INST is valid throughout the bus cycle. INST is activated only during external memory accesses.
EA	Input for memory select (External Access). EA equal to a TTL high causes memory accesses to locations 0000H through 3FFFH to be directed to on-chip ROM/EPROM. EA equal to a TTL low causes accesses to these locations to be directed to off-chip memory.
ALE/ADV	Address Latch Enable or Address Valid output, as selected by CCR. Both pin options provide a latch to demultiplex the address from the address/data bus. When the pin is ADV, it goes inactive high at the end of the bus cycle. ADV can be used as a chip select for external memory. ALE/ADV is activated only during external memory accesses.
RD	Read signal output to external memory. RD is activated only during external memory reads.
WR/WRL	Write and Write Low output to external memory, as selected by the CCR. WR will go low for every external write, while WRL will go low only for external writes when an even byte is being written. WR/WRL is activated only during external memory writes.
BHE/WRH	Bus High Enable or Write High output to external memory, as selected by the CCR. BHE = 0 selects the bank of memory that is connected to the high byte of the data bus. A0 = 0 selects the bank of memory that is connected to the low byte of the data bus. This accesses a 16-bit wide memory can be to the low byte only (A0 = 0, BHE = 1) or the high byte only (A0 = 1, BHE = 0), or both bytes (A0 = 0, BHE = 0). If the WRL function is selected, the pin will go low if the bus cycle is writing to an odd memory location. BHE/WRH is active only during 16-bit external memory write cycles.

7.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings*

Ambient Temperature	
Under Bias	-45°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage On Any Pin to V _{SS}	-0.5V to +7.0V
Power Dissipation	1.5W

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

Operating Conditions

Symbol	Description	Min	Max	Units
T _A	Ambient Temperature Under Bias	-40	+125	°C
V _{CC}	Digital Supply Voltage	4.5	5.50	V
T _{REF}	Analog Supply Voltage	4.5	5.50	V
f _{OSC}	Oscillator Frequency	3.5	10	MHz

NOTE:

ANGND and V_{SS} should be nominally at the same potential.

D.C. Characteristics (Over specified operating conditions)

Symbol	Description	Min	Max	Units	Test Conditions
V _{IL}	Input Low Voltage	-0.5	0.8	V	
V _{IH}	Input High Voltage (except XTAL 1)	0.2 V _{CC} + 0.9	V _{CC} + 0.5	V	
V _{IH1}	Input High Voltage on XTAL 1	0.7 V _{CC}	V _{CC} + 0.5	V	
V _{OL}	Output Low Voltage		0.3 0.45 1.0	V	I _{OL} = 200 µA I _{OL} = 3.2 mA I _{OL} = 7 mA
V _{OH}	Output High Voltage (Standard Outputs)	V _{CC} - 0.3 V _{CC} - 0.7 V _{CC} - 1.5		V	I _{OH} = -200 µA I _{OH} = -3.2 mA I _{OH} = -7 mA
V _{OH1}	Output High Voltage (Quasi-bidirectional Outputs)	V _{CC} - 0.3 V _{CC} - 0.7 V _{CC} - 1.5		V	I _{OH} = -10 µA I _{OH} = -30 µA I _{OH} = -60 µA
I _{LI}	Input Leakage Current (Std. Inputs)		± 15	µA	0 < V _{IN} < V _{CC} - 0.3V
I _{LI1}	Input Leakage Current (Port 0)		± 5	µA	0 < V _{IN} < V _{REF}
I _{TL}	1 to 0 Transition Current (QBD Pins)		-650	µA	V _{IN} = 2.0V
I _{IL}	Logical 0 Input Current (QBD Pins)		-50	µA	V _{IN} = 0.45V
I _{IL1}	Logical 0 Input Current in Reset (ALE, RD, WR, BHE, INST, P2.0)		-500	µA	V _{IN} = 0.45 V

D.C. Characteristics (Over specified operating conditions) (Continued)

Symbol	Description	Min	Max	Units	Test Conditions
I_{CC}	Active Mode Current in Reset		60	mA	$Xtal1 = 10 \text{ MHz}$ $V_{CC} = V_{PP} = V_{REF} = 5.5V$
I_{REF}	A/D Converter Reference Current		5	mA	
I_{idle}	Idle Mode Current		22	mA	
I_{PD}	Powerdown Mode Current		TBD	μA	$V_{CC} = V_{PP} = V_{REF} = 5.5V$
R_{RST}	Reset Pullup Resistor	6K	50K	Ω	
C_S	Pin Capacitance (Any Pin to V_{SS})		10	pF	$f_{TEST} = 1.0 \text{ MHz}$

NOTES:

- QBD (Quasi-bidirectional) pins include Port 1, P2.6 and P2.7.
- Standard Outputs include all bus pins (data and control), HSO pins, PWM/P2.5, CLKOUT, RESET, Ports 3 and 4, TXD/P2.0, and RXD (in serial mode 0). The V_{OH} specification is not valid for RESET. Ports 3 and 4 are open-drain outputs, which will be available on future ROM and EPROM parts.
- Standard Inputs include HSI pins, CDE, EA, READY, BUSWIDTH, NMI, RXD/P2.1, EXTINT/P2.2, T2CLK/P2.3, and T2RST/P2.4.
- Maximum current per pin must be externally limited to the following values if V_{OL} is held above 0.45V or V_{OH} is held below $V_{CC} - 0.7V$:
 I_{OL} on Output pins: 10 mA I_{OH} on quasi-bidirectional pins: self limiting
 I_{OH} on Standard Output pins: 10 mA
- Maximum current per bus pin (data and control) during normal operation is $\pm 3.2 \text{ mA}$.
- During normal (non-transient) conditions the following total current limits apply to each group of pins:
Port 1, P2.6 I_{OL} : 29 mA I_{OH} is self limiting
HS0, P2.0, RXD, RESET I_{OL} : 29 mA I_{OH} : 26 mA
P2.7, P2.5, WR, BHE I_{OL} : 13 mA I_{OH} : 11 mA
AD0-AD15 I_{OL} : 52 mA I_{OH} : 52 mA
RD, ALE, INST-CLKOUT I_{OL} : 13 mA I_{OH} : 13 mA

A.C. Characteristics (Over specified operating conditions)

These are ADVANCED specifications, the parameters may change before Intel releases the product for sale.

Test Conditions: Capacitive load on all pins = 100 pF, Rise and fall times = 10 ns, $f_{OSC} = 10 \text{ MHz}$

The system must meet these specifications to work with the 80C196:

Symbol	Description	Min	Max	Units	Notes
T_{AVYV}	Address Valid to READY Setup		$2T_{OSC} - 45$	ns	
T_{LLYV}	ALE Low to READY Setup		$T_{OSC} - 45$	ns	
T_{YLYH}	NonREADY Time		No upper limit	ns	
T_{CLYX}	READY Hold after CLKOUT Low	0	$T_{OSC} - 5$	ns	(Note 2)
T_{AVGV}	Address Valid to Buswidth Setup		$2T_{OSC} - 45$	ns	
T_{LLGV}	ALE Low to Buswidth Setup		$T_{OSC} - 45$	ns	
T_{CLGX}	Buswidth Hold after CLKOUT Low	0		ns	
T_{AVDV}	Address Valid to Input Data Valid		$3T_{OSC} - 50$	ns	
T_{RLDV}	RD# Active to Input Data Valid		$T_{OSC} - 25$	ns	
T_{CLDV}	CLKOUT Low to Input Data Valid		$T_{OSC} - 45$	ns	(Note 1)
T_{RHDZ}	End of RD# to Input Data Float		$T_{OSC} - 5$	ns	
T_{RXDX}	Data Hold after RD# Inactive	0		ns	

NOTES:

- Typical specification, not guaranteed.
- If max is exceeded, additional wait states will occur.

A.C. Characteristics (Over specified operating conditions) (Continued)

Test Conditions: Capacitive load on all pins = 100 pF, Rise and fall times = 10 ns, $f_{OSC} = 10$ MHz

The 80C196KA will meet these specifications:

Symbol	Description	Min	Max	Units	Notes
FXTAL	Frequency on XTAL1	3.5	10.0	MHz	
T _{OSC}	1/FXTAL	100	286	ns	
T _{XHCH}	XTAL1 High to CLKOUT High or Low	40	110	ns	(Note 1)
T _{CLCL}	CLKOUT Cycle Time	2T _{OSC}		ns	
T _{CHCL}	CLKOUT High Period	T _{OSC} - 15	T _{OSC} + 15	ns	
T _{CLLH}	CLKOUT Falling Edge to CLKOUT Rising	-5	+5	ns	
T _{LLCH}	ALE Falling Edge to CLKOUT Rising	-5	+5	ns	
T _{LHLH}	ALE Cycle Time	4T _{OSC}		ns	
T _{LHLL}	ALE High Period	T _{OSC} - 20	T _{OSC} + 20	ns	
T _{AVLL}	Address Setup to ALE Falling Edge	T _{OSC} - 15		ns	
T _{LLAX}	Address Hold after ALE Falling Edge	T _{OSC} - 15		ns	
T _{LLRL}	ALE Falling Edge to RD Falling Edge	T _{OSC} - 15		ns	
T _{XHRL}	XTAL1 High to RD Falling Edge	35	75	ns	(Note 1)
T _{RLRH}	RD Low Period	T _{OSC} - 15		ns	
T _{RHLH}	RD Rising Edge to ALE Rising Edge	T _{OSC} - 15	T _{OSC} + 15	ns	(Note 2)
T _{LLWL}	ALE Falling Edge to WR Falling Edge	T _{OSC} - 10		ns	
T _{XHWL}	XTAL1 High to WR Falling Edge	55	95	ns	(Note 1)
T _{CLWL}	CLKOUT Low to WR Falling Edge	5	20	ns	(Note 1)
T _{QVWL}	Data Stable to WR Rising Edge	T _{OSC} - 20		ns	
T _{WLWH}	WR Low Period	T _{OSC} - 20		ns	
T _{WHQX}	Data Hold after WR Rising Edge	T _{OSC} - 20		ns	
T _{WHLH}	WR Rising Edge to ALE Rising Edge	T _{OSC} - 20	T _{OSC} + 20	ns	(Note 2)

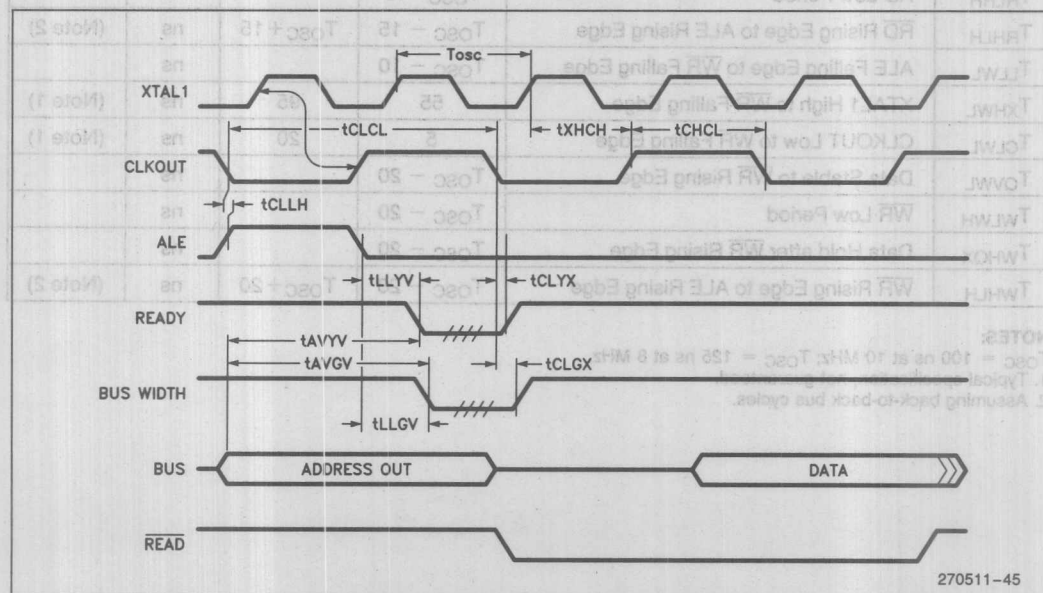
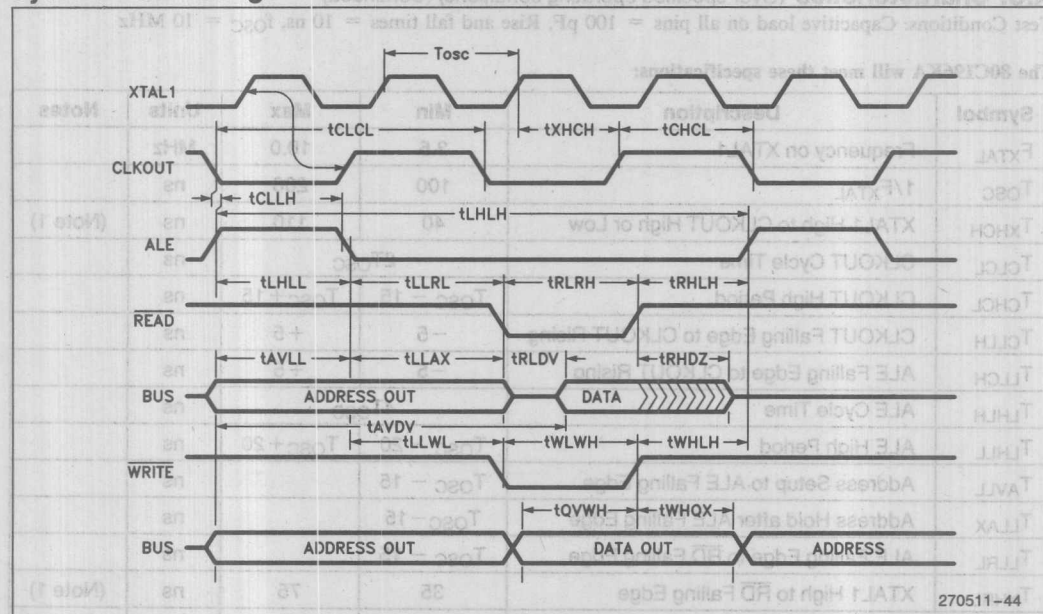
NOTES:

T_{OSC} = 100 ns at 10 MHz; T_{OSC} = 125 ns at 8 MHz.

1. Typical specification, not guaranteed.

2. Assuming back-to-back bus cycles.

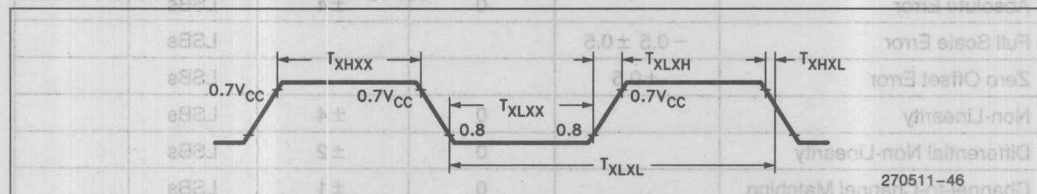
System Bus Timings



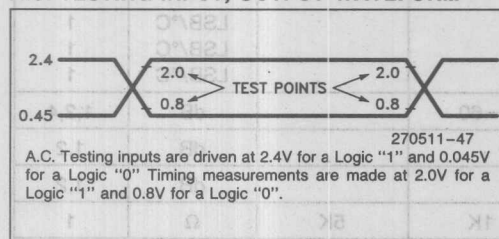
EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
$1/T_{OLOL}$	Oscillator Frequency	3.5	10	MHz
T_{OLOL}	Oscillator Period (T_{OSC})	100	286	ns
T_{OHOX}	High Time	32		ns
T_{OLOX}	Low Time	32		ns
T_{OLOH}	Rise Time		10	ns
T_{OHOL}	Fall Time		10	ns

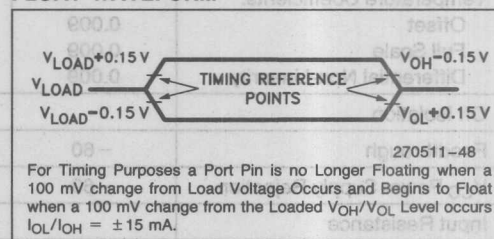
EXTERNAL CLOCK DRIVE WAVEFORMS



A.C. TESTING INPUT, OUTPUT WAVEFORM



FLOAT WAVEFORM



A TO D CHARACTERISTICS

OPERATING CONDITIONS

A/D CONVERTER SPECIFICATIONS

The absolute conversion accuracy is dependent on the accuracy of V_{REF} . The specifications given below assume adherence to the Operating Conditions section of these data sheets. Testing is done at $V_{REF} = 5.12V$.

V_{CC}, V_{REF}	4.5V to 5.5V
$V_{SS}, ANGND$	0V
T_A	-45°C to +125°C
F_{OSC}	3.5 MHz to 10.0 MHz
Test Conditions:	
V_{REF}	5.12V

Parameter	Typical*(1)	Minimum	Maximum	Units**	Notes
Resolution		256 ⁽⁵⁾	1024 10	Levels Bits	
Absolute Error		0	±4	LSBs	
Full Scale Error	-0.5 ± 0.5			LSBs	
Zero Offset Error	± 0.5			LSBs	
Non-Linearity		0	± 4	LSBs	
Differential Non-Linearity		0	± 2	LSBs	
Channel-to-Channel Matching		0	± 1	LSBs	
Repeatability	± 0.25			LSBs1	
Temperature Coefficients:					
Offset	0.009			LSB/°C	1
Full Scale	0.009			LSB/°C	1
Differential Non-Linearity	0.009			LSB/°C	1
Off Isolation		-60		dB	1,2,4
Feedthrough	-60			dB	1,2
V_{CC} Power Supply Rejection	-60			dB	1,2
Input Resistance		1K	5K	Ω	1
D.C. Input Leakage		0	5.0	μA	

NOTES:

* These values are expected for most parts at 25°C.

**An "LSB", as used here, has a value of approximately 5 mV.

- These values are not tested in production and are based on theoretical estimates and laboratory tests.
- DC to 100 KHz.
- For starting the A/D with an HSO Command.
- Multiplexer Break-Before-Make Guaranteed.
- See functional deviations list.

A/D GLOSSARY OF TERMS

ABSOLUTE ERROR—The maximum difference between corresponding actual and ideal code transitions. Absolute Error accounts for all deviations of an actual converter from an ideal converter.

ACTUAL CHARACTERISTIC—The characteristic of an actual converter. The characteristic of a given converter may vary over temperature, supply voltage, and frequency conditions. An actual characteristic rarely has ideal first and last transition locations or ideal code widths. It may even vary over multiple conversions under the same conditions.

BREAK-BEFORE-MAKE—The property of a multiplexer which guarantees that a previously selected channel will be deselected before a new channel is selected. (e.g. the converter will not short inputs together.)

CHANNEL-TO-CHANNEL MATCHING—The difference between corresponding code transitions of actual characteristics taken from different channels under the same temperature, voltage and frequency conditions.

CHARACTERISTIC—A graph of input voltage versus the resultant output code for an A/D converter. It describes the transfer function of the A/D converter.

CODE—The digital value output by the converter.

CODE CENTER—The voltage corresponding to the midpoint between two adjacent code transitions.

CODE TRANSITION—The point at which the converter changes from an output code of Q , to a code of $Q + 1$. The input voltage corresponding to a code transition is defined to be that voltage which is equally likely to produce either of two adjacent codes.

CODE WIDTH—The voltage corresponding to the difference between two adjacent code transitions.

CROSSTALK—See "Off-Isolation".

D.C. INPUT LEAKAGE—Leakage current to ground from an analog input pin.

DIFFERENTIAL NON-LINEARITY—The difference between the ideal and actual code widths of the terminal based characteristic.

FEEDTHROUGH—Attenuation of a voltage applied on the selected channel of the A/D Converter after the sample window closes.

FULL SCALE ERROR—The difference between the expected and actual input voltage corresponding to the full scale code transition.

IDEAL CHARACTERISTIC—A characteristic with its first code transition at $V_{IN} = 0.5 \text{ LSB}$, its last code transition at $V_{IN} = (V_{REF} - 1.5 \text{ LSB})$ and all code widths equal to one LSB.

INPUT RESISTANCE—The effective series resistance from the analog input pin to the sample capacitor.

LSB—Least Significant Bit: The voltage corresponding to the full scale voltage divided by 2^n , where n is the number of bits of resolution of the converter. For an 8-bit converter with a reference voltage of 5.12V, one LSB is 20 mV. Note that this is different than digital LSBs, since an uncertainty of two LSB, when referring to an A/D converter, equals 40 mV. (This has been confused with an uncertainty of two digital bits, which would mean four counts, or 80 mV.)

MONOTONIC—The property of successive approximation converters which guarantees that increasing input voltages produce adjacent codes of increasing value, and that decreasing input voltages produce adjacent codes of decreasing value.

NO MISSED CODES—For each and every output code, there exists a unique input voltage range which produces that code only.

NON-LINEARITY—The maximum deviation of code transitions of the terminal based characteristic from the corresponding code transitions of the ideal characteristic.

OFF-ISOLATION—Attenuation of a voltage applied on a deselected channel of the A/D converter. (Also referred to as Crosstalk.)

REPEATABILITY—The difference between corresponding code transitions from different actual characteristics taken from the same converter on the same channel at the same temperature, voltage and frequency conditions.

between. Also defines the number of useful bits of information which the converter can return.

SAMPLE DELAY—The delay from receiving the start conversion signal to when the sample window opens.

SAMPLE DELAY UNCERTAINTY—The variation in the sample delay.

SAMPLE TIME—The time that the sample window is open.

SAMPLE TIME UNCERTAINTY—The variation in the sample time.

SAMPLE WINDOW—Begins when the sample capacitor is attached to a selected channel and ends when the sample capacitor is disconnected from the selected channel.

SUCCESSIVE APPROXIMATION—An A/D conversion method which uses a binary search to arrive at the best digital representation of an analog input.

TEMPERATURE COEFFICIENTS—Change in the stated variable per degree centigrade temperature change. Temperature coefficients are added to the typical values of a specification to see the effect of temperature drift.

TERMINAL BASED CHARACTERISTIC—An actual characteristic which has been rotated and translated to remove zero offset and full scale error.

V_{CC} REJECTION—Attenuation of noise on the V_{CC} line to the A/D converter.

ZERO OFFSET—The difference between the expected and actual input voltage corresponding to the first code transition.

The 80C196KA has the following problems. We are working on, or have already defined, silicon fixes for all these problems.

1. Byte shifts on odd addresses do not work properly (SHRB and SHLB). Byte shifts can be done on even addresses, and word and long shifts work correctly.
2. The Unsigned Divide operations (Byte and Word), may result in a quotient that is one count larger than the correct value (DIVU and DIVUB). This can only occur if the most significant bit of the divisor is a one. The problem will not always occur if the MSB is one, and determining if the problem will occur or not is very difficult.
3. The current in the power down mode is on the order of 1 milliamp.
4. The PUSHA instruction works properly with internal stack. When external stack is used, the PUSHA instruction will cause the data to be written into the location pointed to by the lower byte of the stack pointer. Since the PUSHA instruction is simply a fast way of doing a PUSHF, and pushing PSR/IMASK1 and clearing IMASK1, a macro can be written to work around this problem.
5. The A/D converter differential non-linearity error becomes larger as V_{in} approaches V_{ref}. This results in the potential for missed codes at 10-bit resolution.
6. The reset pin must have a rise time less than 4 state times. An External Schmitt trigger reset circuit is recommended. A capacitor only or RC circuit directly connected to the pin will not work reliably. If a bad reset occurs, the chip will lock-up. A good reset will cause the part to work correctly; the chip does not have to be powered on and off.

NOTE:

Instruction bugs 1, 2, and 3 may prevent high level language compilers from generating code which works correctly. If a problem is suspected, generate an assembler code output of the high level language and examine the listing for the above instructions. If any of the instructions are present, the code may have to be rewritten.

**MCS®-96 Data Sheets,
Application Notes, Article
Reprints and
Development Support Tools**

19

MCS®-96 Data Sheets,
Application Notes, Article
Reprints and
Development Support Tools

19

MCS®-96 **809X-90, 839X-90** **AUTOMOTIVE**

- 839X: an 809X with 8 Kbytes of On-Chip ROM
- High Speed Pulse I/O
- 10-Bit A/D Converter
- 6.25 μ s 16 x 16 Multiply
- 6.25 μ s 32/16 Divide
- 8 Interrupt Sources
- Pulse-Width Modulated Output
- 232 Byte Register File
- Memory-to-Memory Architecture
- Full Duplex Serial Port
- Five 8-Bit I/O Ports
- Watchdog Timer
- Four 16-Bit Software Timers

The MCS®-96 family of 16-bit microcontrollers consists of many members, all of which are designed for high-speed control functions. Members with the "-90" suffix are described in this data sheet.

The CPU supports bit, byte, and word operations. 32-bit double-words are supported for a subset of the instruction set. With a 12 MHz input frequency the 8096 can do a 16-bit addition in 1.0 μ s and a 16 x 16-bit multiply or 32/16-bit divide in 6.25 μ s. Instruction execution times average 1 to 2 μ s in typical applications.

Four high-speed trigger inputs are provided to record the times at which external events occur. Six high-speed pulse generator outputs are provided to trigger external events at present times. The high-speed output unit can simultaneously perform timer functions. Up to four such 16-bit Software Timers can be in operation at once.

An on-chip A/D Converter converts up to 8 analog input channels to 10-bit digital values. This feature is only available on the 8095-90/8395-90 and 8097-90/8397-90.

Also provided on-chip are a serial port, a Watchdog Timer, and a pulse-width modulated output signal.

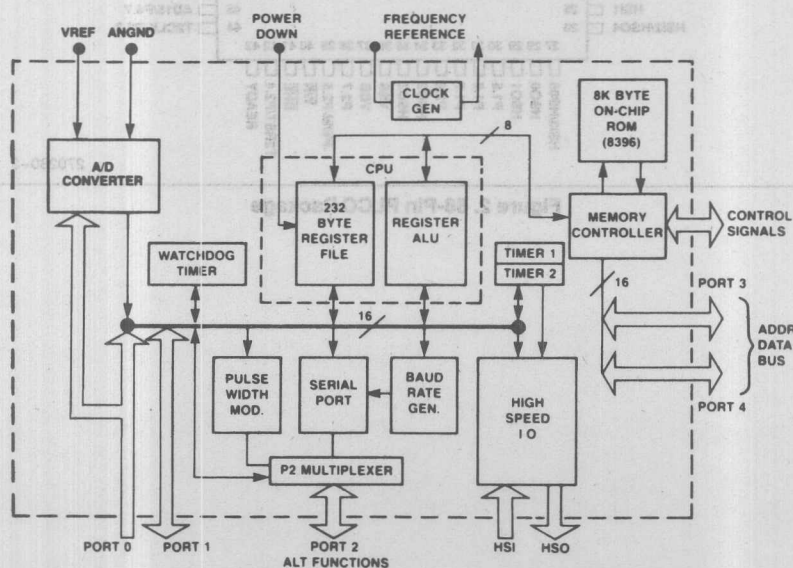


Figure 1. Block Diagram

270280-1

Figure 1 shows a block diagram of the MCS-96 parts, generally referred to as the 8096. The 8096 is available in 48-pin and 68-pin packages, with and without A/D, and with and without on-chip ROM the MCS-96 numbering system is shown below:

Options		68-Pin
Digital I/O	ROMLESS	8096-90
	ROM	8396-90
Analog and Digital I/O	ROMLESS	8097-90
	ROM	8397-90

Figures 2 and 3 show the pinouts for the 68-pin packages. The 68-pin version comes in a Plastic Leaded Chip Carrier, a Plastic Flat Pack and a Pin Grid Array.

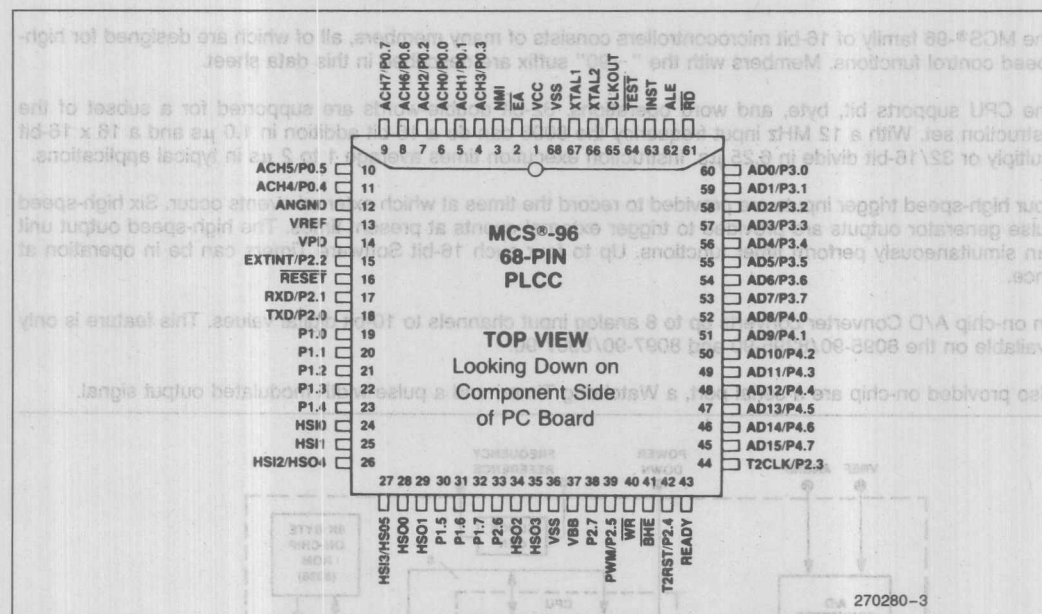


Figure 2. 68-Pin PLCC Package

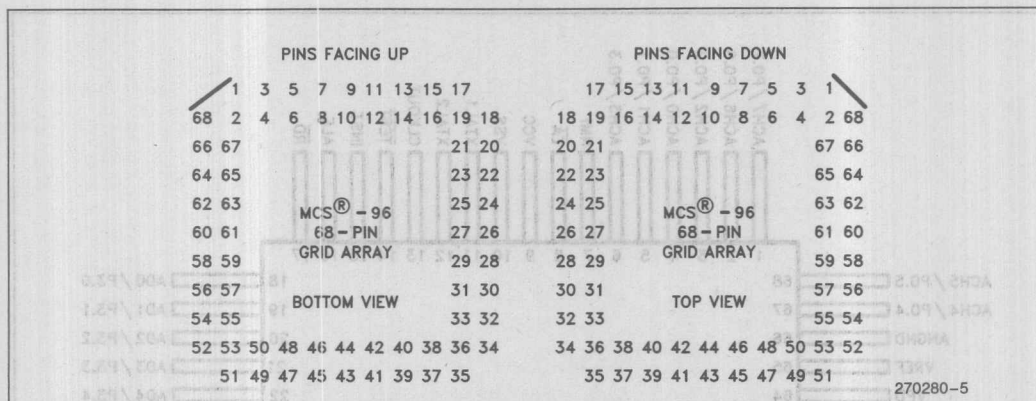


Figure 4. Pin Grid Array

NOTE:

1. When the pin grid array package is mounted on the PC board, the pins are numbered counterclockwise as seen from the component side of the board, just like the flat pack when its mounted in the contactor. Consequently, the PC board layout for pin grid array is compatible with the flat pack in a contactor except for the footprint size. The pin functions (from 1 to 68) on both packages are identical. Refer to Intel's Embedded Controller handbook for mechanical dimensions on these packages.

PGA	PLCC	Description	PGA	PLCC	Description	PGA	PLCC	Description
1	9	ACH7/P0.7	24	54	AD6/P3.6	47	31	P1.6
2	8	ACH6/P0.6	25	53	AD7/P3.7	48	30	P1.5
3	7	ACH2/P0.2	26	52	AD8/P4.0	49	29	HSO.1
4	6	ACH0/P0.0	27	51	AD9/P4.1	50	28	HSO.0
5	5	ACH1/P0.1	28	50	AD10/P4.2	51	27	HSO.5/HSI.3
6	4	ACH3/P0.3	29	49	AD11/P4.3	52	26	HSO.4/HSI.2
7	3	NMI	30	48	AD12/P4.4	53	25	HSI.1
8	2	EA	31	47	AD13/P4.5	54	24	HSI.0
9	1	V _{CC}	32	46	AD14/P4.6	55	23	P1.4
10	68	V _{SS}	33	45	AD15/P4.7	56	22	P1.3
11	67	XTAL1	34	44	T2CLK/P2.3	57	21	P1.2
12	66	XTAL2	35	43	READY	58	20	P1.1
13	65	CLKOUT	36	42	T2RST/P2.4	59	19	P1.0
14	64	TEST	37	41	BHE	60	18	TXD/P2.0
15	63	INST	38	40	WR	61	17	RXD/P2.1
16	62	ALE	39	39	PWM/P2.5	62	16	RESET
17	61	RD	40	38	P2.7	63	15	EXTINT/P2.2
18	60	AD0/P3.0	41	37	V _{BB}	64	14	V _{PD}
19	59	AD1/P3.1	42	36	V _{SS}	65	13	V _{REF}
20	58	AD1/P3.2	43	35	HSO.3	66	12	ANGND
21	57	AD3/P3.3	44	34	HSO.2	67	11	ACH4/P0.4
22	56	AD4/P3.4	45	33	P2.6	68	10	ACH5/P0.5
23	55	AD5/P3.5	46	32	P1.7			

FUNCTIONAL OVERVIEW

The following section is an overview of the 8096, the generic part number used to refer to the entire MCS-96 product family. Additional information is available in the Embedded Controller Handbook, order number 210918.

CPU Architecture

The 8096 has 64 Kbyte addressability and uses the same address space for both program and data memory, except in the address range from 00H through 0FFH. Data fetches in this range are always to the Register File, while instruction fetches from these locations are directed to external memory. (Locations 00H through 0FFH in external memory are reserved for Intel development systems.)

Within the Register File, locations 00H through 17H are register mapped I/O control registers, also re-

ferred to as Special Function Registers (SFRs). The rest of the Register File (018H through 0FFH) contains 232 bytes of RAM, which can be referenced as bytes, words, or double-words. This register space allows the user to keep the most frequently-used variables in on-chip RAM, which can be accessed faster than external memory. Locations 0F0H through 0FFH can be preserved during power down if power is applied to the V_{PD} pin.

Outside of the register file, program memory, data memory, and peripherals can be intermixed. The addresses with special significance are:

0000H—0017H	Register-mapped I/O (SFRs)
0018H—0019H	Stack Pointer
1FFE—1FFFH	Ports 3 and 4
2000H—2011H	Interrupt Vectors
2012H—207FH	Factory Test Code
2080H	Reset Location

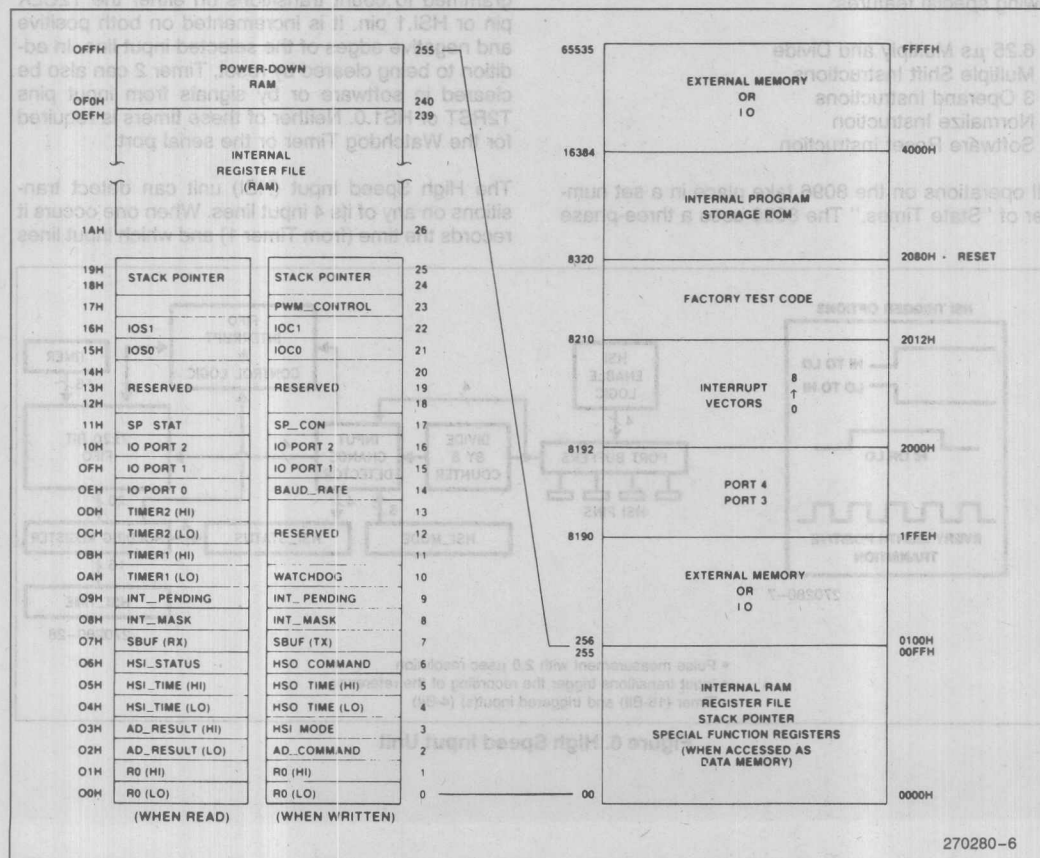


Figure 5. Memory Map

The 839X carries 8 Kbytes of on-chip ROM, occupying addresses 2000H through 3FFFH. Instruction or data fetches from these addresses access the on-chip ROM if the \overline{EA} pin is externally held at a logical 1. If the \overline{EA} pin is at a logical 0 these addresses access off-chip memory.

A memory map for the MCS-96 product family is shown in Figure 5.

The RALU (Register/ALU) section consists of a 17-bit ALU, the Program Status Word, the Program Counter, and several temporary registers. A key feature of the 8096 is that it does not use an accumulator. Rather, it operates directly on any register in the Register File. Being able to operate directly on data in the Register File without having to move it into and out of an accumulator results in a significant improvement in execution speed.

In addition to the normal arithmetic and logical functions, the MCS-96 instruction set provides the following special features:

- 6.25 μ s Multiply and Divide
- Multiple Shift Instructions
- 3 Operand Instructions
- Normalize Instruction
- Software Reset Instruction

All operations on the 8096 take place in a set number of "State Times." The 8096 uses a three-phase

internal clock, so each state time is 3 oscillator periods. With a 12 MHz clock, each state time requires 0.25 microseconds.

High Speed I/O Unit (HSIO)

The HSIO unit consists of the High Speed Input Unit (HSI), the High Speed Output Unit (HSO), one counter and one timer. "High Speed" denotes that the units can perform functions related to the timers without CPU intervention. The HSI records times when events occur and the HSO triggers events at preprogrammed times.

All actions within the HSIO unit are synchronized to the timers. The two 16-bit timer/counter registers in the HSIO unit are cleared on chip reset and can be programmed to generate an interrupt on overflow. The Timer 1 register is automatically incremented every 8 state times (every 2.0 microseconds, with a 12 MHz clock). The Timer 2 register can be programmed to count transitions on either the T2CLK pin or HSI.1 pin. It is incremented on both positive and negative edges of the selected input line. In addition to being cleared by reset, Timer 2 can also be cleared in software or by signals from input pins T2RST or HS1.0. Neither of these timers is required for the Watchdog Timer or the serial port.

The High Speed Input (HSI) unit can detect transitions on any of its 4 input lines. When one occurs it records the time (from Timer 1) and which input lines

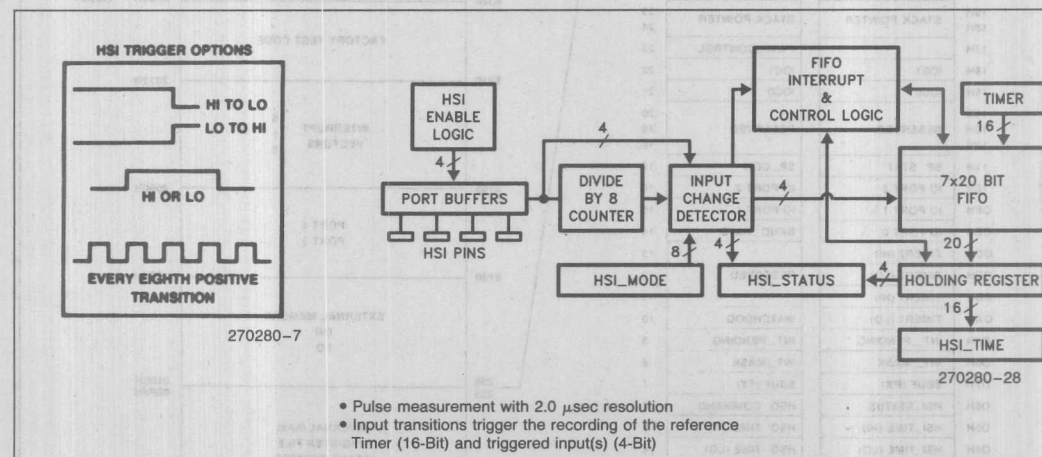


Figure 6. High Speed Input Unit

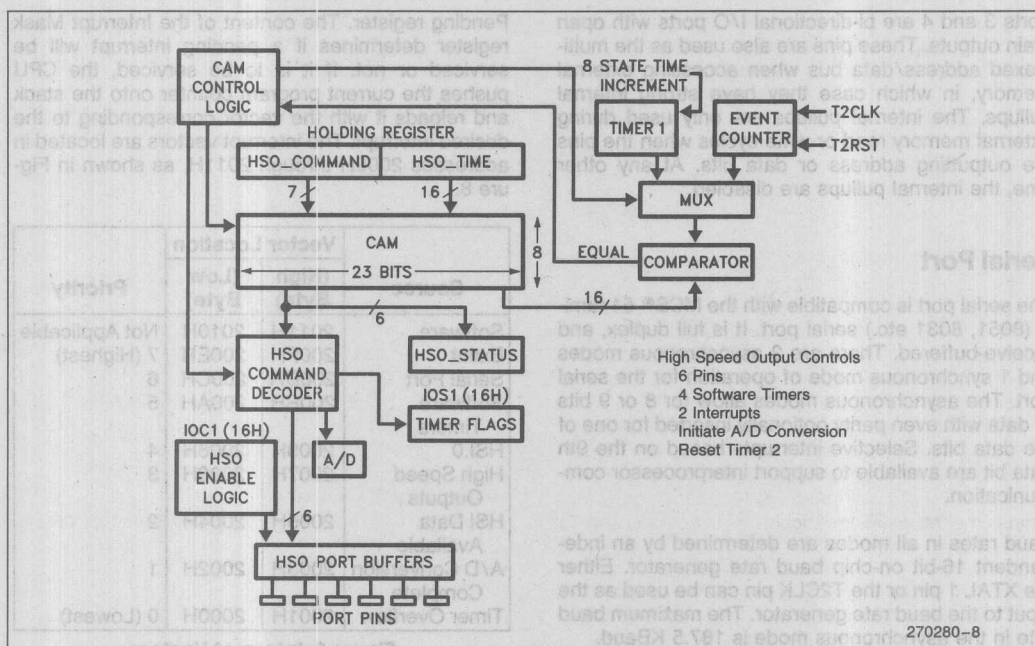


Figure 7. High Speed Output Unit

made the transition. This information is recorded with 2 microsecond resolution and stored in an 8-level FIFO. The unit can be programmed to look for four types of events, as shown in Figure 6. It can activate the HSI Data Available interrupt either when the Holding Registers is loaded or the 6th FIFO entry has been made. Each input line can be individually enabled or disabled to the HSI unit by software.

The High Speed Output (HSO) unit is shown in Figure 7. It can be programmed to set or clear any of its 6 output lines, reset Timer 2, trigger an A/D conversion, or set one of 4 Software Timers flags at a programmed time. An interrupt can be enabled for any of these events. Either Timer 1 or Timer 2 can be referenced for the programmed time value and up to 8 commands for preset actions can be stored in the CAM (Content Addressable Memory) file at any one time. As each action is carried out at its preset time that command is removed from the CAM making space for another command. HSO.4 and HSO.5 are shared with the HSI unit as HSI.2 and HSI.3, and can be individually enabled or disabled as outputs.

Standard I/O Ports

There are 5 8-bit I/O ports on the 8096 in addition to the High Speed I/O lines.

Port 0 is an input-only port which shares its pins with the analog inputs to the A/D Converter. The port

can be read digitally and/or, by writing to the A/D Command Register, one of the lines can be selected as the input to the A/D Converter.

Port 1 is a quasi-bidirectional I/O port. "Quasi-bidirectional" means the port pin has a weak internal pullup that is always active and an internal pulldown which can either be on (to output a 0) or off (to output a 1). This configuration allows the pin to be used as either an input or an output without using a data direction register. In parallel with the weak internal pullup, is a much stronger internal pulldown that is activated for one state time when the pin is internally driven from 0 to 1. This is done to speed up the 0-to-1 transition time.

Port 2 is multi-functional port. Two of the pins are quasi-bidirectional while the remaining six are shared with other functions in the 8096, as shown below:

Port	Function	Alternate Function
P2.0	output	TXD (serial port transmit)
P2.1	input	RXD (serial port receive)
P2.2	input	EXTINT (external interrupt)
P2.3	input	T2CLK (Timer 2 clock)
P2.4	input	T2RST (Timer 2 reset)
P2.5	output	PWM (pulse-width modulation)

Ports 3 and 4 are bi-directional I/O ports with open drain outputs. These pins are also used as the multiplexed address/data bus when accessing external memory, in which case they have strong internal pullups. The internal pullups are only used during external memory read or write cycles when the pins are outputting address or data bits. At any other time, the internal pullups are disabled.

Serial Port

The serial port is compatible with the MCS®-51 family (8051, 8031 etc.) serial port. It is full duplex, and receive-buffered. There are 3 asynchronous modes and 1 synchronous mode of operation for the serial port. The asynchronous modes allow for 8 or 9 bits of data with even parity optionally inserted for one of the data bits. Selective interrupts based on the 9th data bit are available to support interprocessor communication.

Baud rates in all modes are determined by an independent 16-bit on-chip baud rate generator. Either the XTAL 1 pin or the T2CLK pin can be used as the input to the baud rate generator. The maximum baud rate in the asynchronous mode is 187.5 KBaud.

Pulse Width Modulator (PWM)

The PWM output shares a pin with port bit P2.5. When the PWM output is selected, this pin outputs a pulse train having a fixed period of 256 state times, and a programmable width of 0 to 255 state times. The width is programmed by loading the desired value, in state times, to the PWM Control Register.

A/D Converter

The analog-to-digital converter is a 10-bit, successive approximation converter. It has a fixed conversion time of 168 state times, (42 microseconds with a 12 MHz clock). The analog input must be in the range of 0 to VREF (normally, VREF = 5V). This input can be selected from 8 analog input lines, which connect to the same pins as Port 0. A conversion can be initiated either by setting a control bit in the A/D Command register, or by programming the HSO unit to trigger the conversion at some specified time.

Interrupts

The 8096 has 20 interrupt sources which vector through 8 locations. A 0-to-1 transition from any of the sources sets a corresponding bit in the Interrupt

Pending register. The content of the Interrupt Mask register determines if a pending interrupt will be serviced or not. If it is to be serviced, the CPU pushes the current program counter onto the stack and reloads it with the vector corresponding to the desired interrupt. The interrupt vectors are located in addresses 2000H through 2011H, as shown in Figure 8.

Source	Vector Location		Priority
	(High Byte)	(Low Byte)	
Software	2011H	2010H	Not Applicable 7 (Highest)
Extint	200FH	200EH	
Serial Port	200DH	200CH	
Software	200BH	200AH	
Timers			5
HSI.0	2009H	2008H	4
High Speed Outputs	2007H	2006H	3
HSI Data Available	2005H	2004H	2
A/D Conversion Complete	2003H	2002H	1
Timer Overflow	2001H	2000H	0 (Lowest)

Figure 8. Interrupt Vectors

At the end of the terminal routine the RET instruction pops the program counter from the stack and execution continues where it left off. It is not necessary to store and replace registers during interrupt routines as each routine can be set up to use a different section of the register file. This feature of the architecture provides for very fast context switching.

While the 8096 has a single priority level in the sense that any interrupt may itself be interrupted, a priority structure exists for resolving simultaneously pending interrupts, as indicated in Figure 9. Since the interrupt pending and interrupt mask registers can be manipulated in software, it is possible to dynamically alter the interrupt priorities to suit the users' software.

Watchdog Timer

The Watchdog Timer is a 16-bit counter which, once started, is incremented every state time. After 16 milliseconds, if not cleared, it will overflow, pulling down the RESET pin for two state times, causing the system to be reinitialized. This feature is provided as a means of graceful recovery from a software upset. The counter must be cleared by the software before it overflows, or else the system assumes an upset has occurred and activates RESET.

PIN DESCRIPTION

V_{CC}

Main supply voltage (5V).

V_{SS}

Digital circuit ground (0V).

V_{PD}

RAM standby supply voltage (5V). This voltage must be present during normal operation. In a Power Down condition (i.e., V_{CC} drops to zero), if RESET is activated before V_{CC} drops below spec and V_{PD} continues to be held within spec, the top 16 bytes in the Register File will retain their contents. RESET must be held low during the Power Down and should not be brought high until V_{CC} is within spec and the oscillator has stabilized.

V_{REF}

Reference voltage for the A/D converter (5V). V_{REF} is also the supply voltage to the analog portion of the A/D converter and the logic used to read Port 0 as digital I/O.

ANGND

Reference ground for the A/D converter. Should be held at nominally the same potential as V_{SS}.

V_{BB}

Substrate voltage from the on-chip back-bias generator. This pin should be connected to ANGND through a 0.01 μ f capacitor (and not connected to anything else).

XTAL1

Input of the oscillator inverter and of the internal clock generator.

XTAL2

Output of the oscillator inverter.

CLKOUT

Output of the internal clock generator. The frequency of CLKOUT is $\frac{1}{3}$ the oscillator frequency. It has a 33% duty cycle.

RESET

Reset input to the chip. Input low for at least 2 state times to reset the chip. The subsequent low-to-high transition re-synchronizes CLKOUT and commences a 10-state-time sequence in which the PSW is cleared and a jump to address 2080H is executed. Input high for normal operation. RESET has an internal pullup.

TEST

Input low enables a factory test mode. The user should tie this pin to V_{CC} for normal operation.

NMI

A positive transition clears the watchdog timer, and causes a vector to external memory location 0000H. External memory from 00H through 0FFH is reserved for Intel development systems.

INST

Output high during an external memory read indicates the read is an instruction fetch. INST needs to be latched on falling edge of ALE.

EA

Input for memory select (External Access). EA = 1 causes memory accesses to locations 2000H through 3FFFH to be directed to on-chip ROM. EA = 0 causes accesses to these locations to be directed to off-chip memory. EA has an internal pull-down, so it goes to 0 unless driven to 1. EA is not latched internally during RESET.

ALE

Address Latch Enable output. ALE is activated only during external memory accesses. It is used to latch the address from the multiplexed address/data bus, and is placed in a low state during RESET.

RD

Read signal output to external memory. RD is activated only during external memory reads.

WR

Write signal output to external memory. WR is activated only during external memory writes.

BHE

Bus High Enable signal output to external memory. $\overline{\text{BHE}} = 0$ selects the bank of memory that is connected to the high byte of the data bus. $\text{A0} = 0$ selects the bank of memory that is connected to the low byte of the data bus. Thus accesses to a 16-bit wide memory can be to the low byte only ($\text{A0} = 0$, $\overline{\text{BHE}} = 1$), to the high byte only ($\text{A0} = 1$, $\overline{\text{BHE}} = 0$), or to both bytes ($\text{A0} = 0$, $\overline{\text{BHE}} = 0$). $\overline{\text{BHE}}$ is activated only when required during accesses to external memory. $\overline{\text{BHE}}$ can be ignored during read operations. This pin must be latched on the falling edge of ALE.

READY

The READY input is used to lengthen external memory bus cycles, for interfacing to slow or dynamic memory, or for bus sharing. If the pin is high, CPU operation continues in a normal manner. If the pin is low prior to the rising edge of CLKOUT, after ALE the Memory Controller goes into a wait mode until the next negative transition in CLKOUT after ALE occurs with READY high. The bus cycle can be lengthened by up to 1 μs . When the external memory bus is not being used, READY has no effect. READY has a weak internal pullup, so it goes to 1 unless externally pulled low.

HSI

Inputs to High Speed Input Unit. Four HSI pins are available: HSI.0, HSI.1, HSI.2, and HSI.3. Two of them (HSI.2 and HSI.3) are shared with the HSO Unit.

HSO

Outputs from High Speed Output Unit. Six HSO pins are available: HSO.0, HSO.1, HSO.2, HSO.3, HSO.4, and HSO.5. Two of them (HSO.4 and HSO.5) are shared with the HSI Unit.

Port 0

8-bit high impedance input-only port. These pins can be used as digital inputs and/or as analog inputs to the on-chip A/D converter.

Port 1

8-bit quasi-bidirectional I/O port.

Port 2

8-bit multi-functional port. Six of its pins are shared with other functions in the 8096, the remaining 2 are quasi-bidirectional.

Ports 3 and 4

8-bit bi-directional I/O ports with open drain outputs. These pins are shared with the multiplexed address/data bus which has strong internal pullups.

INSTRUCTION SET

The 8096 instruction set makes use of six addressing modes as described below:

DIRECT—The operand is specified by an 8-bit address field in the instruction. The operand must be in the Register File or SFR space (locations 0000H through 00FFH).

IMMEDIATE—The operand itself follows the opcode in the instruction stream as immediate data. The immediate data can be either 8-bits or 16-bits as required by the opcode.

INDIRECT—An 8-bit address field in the instruction gives the address of a word register in the Register File which contains the 16-bit address of the operand. The operand can be anywhere in memory.

INDIRECT WITH AUTO-INCREMENT—Same as Indirect, except that, after the operand is referenced, the word register that contains the operand's address is incremented by 1 if the operand is a byte, or by 2 if the operand is a word.

INDEXED—The instruction contains an 8-bit address field and either an 8-bit or a 16-bit displacement field. The 8-bit address field gives the address of a word register in the Register File which contains a 16-bit base address. The 8- or 16-bit displacement field contains a signed displacement that will be added to the base address to produce the address of the operand. The operand can be anywhere in memory.

The 8096 contains a Zero Register at word address 0000H (and which contains 0000H). This register is available for performing comparisons and for use as a base register in indexed addressing. This effectively provides direct addressing to all 64K of memory.

In the 8096, the Stack Pointer is at word address 0018H in the Register File. If the 8-bit address field in an indexed instruction contains 18H, the Stack Pointer becomes the base register. This allows direct accessing of variables in the stack.

The following tables list the MCS-96 instructions, their opcodes, and execution times.

Instruction Summary

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	$D \leftarrow D + A$	✓	✓	✓	✓	↑	—	
ADD/ADDB	3	$D \leftarrow B + A$	✓	✓	✓	✓	↑	—	
ADDC/ADDCB	2	$D \leftarrow D + A + C$	↓	✓	✓	✓	↑	—	
SUB/SUBB	2	$D \leftarrow D - A$	✓	✓	✓	✓	↑	—	
SUB/SUBB	3	$D \leftarrow B - A$	✓	✓	✓	✓	↑	—	
SUBC/SUBCB	2	$D \leftarrow D - A + C - 1$	↓	✓	✓	✓	↑	—	
CMP/CMPB	2	$D - A$	✓	✓	✓	✓	↑	—	
MUL/MULU	2	$D, D + 2 \leftarrow D * A$	—	—	—	—	—	?	2
MUL/MULU	3	$D, D + 2 \leftarrow B * A$	—	—	—	—	—	?	2
MULB/MULUB	2	$D, D + 1 \leftarrow D * A$	—	—	—	—	—	?	3
MULB/MULUB	3	$D, D + 1 \leftarrow B * A$	—	—	—	—	—	?	3
DIVU	2	$D \leftarrow (D, D + 2) / A, D + 2 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	2
DIVUB	2	$D \leftarrow (D, D + 1) / A, D + 1 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	3
DIV	2	$D \leftarrow (D, D + 2) / A, D + 2 \leftarrow \text{remainder}$	—	—	—	?	↑	—	2
DIVB	2	$D \leftarrow (D, D + 1) / A, D + 1 \leftarrow \text{remainder}$	—	—	—	?	↑	—	3
AND/ANDB	2	$D \leftarrow D \text{ and } A$	✓	✓	0	0	—	—	
AND/ANDB	3	$D \leftarrow B \text{ and } A$	✓	✓	0	0	—	—	
OR/ORB	2	$D \leftarrow D \text{ or } A$	✓	✓	0	0	—	—	
XOR/XORB	2	$D \leftarrow D \text{ (excl. or) } A$	✓	✓	0	0	—	—	
LD/LDB	2	$D \leftarrow A$	—	—	—	—	—	—	
ST/STB	2	$A \leftarrow D$	—	—	—	—	—	—	
LDBSE	2	$D \leftarrow A; D + 1 \leftarrow \text{SIGN}(A)$	—	—	—	—	—	—	3, 4
LDBZE	2	$D \leftarrow A; D + 1 \leftarrow 0$	—	—	—	—	—	—	3, 4
PUSH	1	$SP \leftarrow SP - 2; (SP) \leftarrow A$	—	—	—	—	—	—	
POP	1	$A \leftarrow (SP); SP \leftarrow SP + 2$	—	—	—	—	—	—	
PUSHF	0	$SP \leftarrow SP - 2; (SP) \leftarrow \text{PSW};$ $\text{PSW} \leftarrow 0000\text{H}$	0	0	0	0	0	0	
POPF	0	$\text{PSW} \leftarrow (SP); SP \leftarrow SP + 2; I \leftarrow \text{PSW}$	✓	✓	✓	✓	✓	✓	
SJMP	1	$PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LJMP	1	$PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5
BR (indirect)	1	$PC \leftarrow (A)$	—	—	—	—	—	—	
SCALL	1	$SP \leftarrow SP - 2; (SP) \leftarrow PC;$ $PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LCALL	1	$SP \leftarrow SP - 2; (SP) \leftarrow PC;$ $PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5
RET	0	$PC \leftarrow (SP); SP \leftarrow SP + 2$	—	—	—	—	—	—	
J (conditional)	1	$PC \leftarrow PC + 8\text{-bit offset (if taken)}$	—	—	—	—	—	—	5
JC	1	Jump if C = 1	—	—	—	—	—	—	5
JNC	1	Jump if C = 0	—	—	—	—	—	—	5
JE	1	Jump if Z = 1	—	—	—	—	—	—	5

NOTES:

1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
2. D, D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D, D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to a word.
5. Offset is a 2's complement number.

Instruction Summary (Continued)

Mnemonic	Oper- ands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
JNE	1	Jump if Z = 0	—	—	—	—	—	—	5
JGE	1	Jump if N = 0	—	—	—	—	—	—	5
JLT	1	Jump if N = 1	—	—	—	—	—	—	5
JGT	1	Jump if N = 0 and Z = 0	—	—	—	—	—	—	5
JLE	1	Jump if N = 1 or Z = 1	—	—	—	—	—	—	5
JH	1	Jump if C = 1 and Z = 0	—	—	—	—	—	—	5
JNH	1	Jump if C = 0 or Z = 1	—	—	—	—	—	—	5
JV	1	Jump if V = 1	—	—	—	—	—	—	5
JNV	1	Jump if V = 0	—	—	—	—	—	—	5
JVT	1	Jump if VT = 1; Clear VT	—	—	—	—	0	—	5
JNVT	1	Jump if VT = 0; Clear VT	—	—	—	—	0	—	5
JST	1	Jump if ST = 1	—	—	—	—	—	—	5
JNST	1	Jump if ST = 0	—	—	—	—	—	—	5
JBS	3	Jump if Specified Bit = 1	—	—	—	—	—	—	5, 6
JBC	3	Jump if Specified Bit = 0	—	—	—	—	—	—	5, 6
DJNZ	1	D ← D - 1; if D ≠ 0 then PC ← PC + 8-bit offset	—	—	—	—	—	—	5
DEC/DECB	1	D ← D - 1	✓	✓	✓	✓	↑	—	—
NEG/NEGB	1	D ← 0 - D	✓	✓	✓	✓	↑	—	—
INC/INCB	1	D ← D + 1	✓	✓	✓	✓	↑	—	—
EXT	1	D ← D; D + 2 ← Sign (D)	✓	✓	0	0	—	—	2
EXTB	1	D ← D; D + 1 ← Sign (D)	✓	✓	0	0	—	—	3
NOT/NOTB	1	D ← Logical Not (D)	✓	✓	0	0	—	—	—
CLR/CLRB	1	D ← 0	1	0	0	0	—	—	—
SHL/SHLB/SHLL	2	C ← msb ——— lsb ← 0	✓	?	✓	✓	↑	—	7
SHR/SHRB/SHRL	2	0 → msb ——— lsb → C	✓	?	✓	0	—	✓	7
SHRA/SHRAB/SHRAL	2	msb → msb ——— lsb → C	✓	✓	✓	0	—	✓	7
SETC	0	C ← 1	—	—	1	—	—	—	—
CLRC	0	C ← 0	—	—	0	—	—	—	—
CLRVT	0	VT ← 0	—	—	—	—	0	—	—
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	—	—	—	—	—	—	—
EI	0	Enable All Interrupts (I ← 1)	—	—	—	—	—	—	—
NOP	0	PC ← PC + 1	—	—	—	—	—	—	—
SKIP	0	PC ← PC + 2	—	—	—	—	—	—	—
NORML	2	Left Shift Till msb = 1; D ← shift count	✓	?	0	—	—	—	7
TRAP	0	SP ← SP - 2; (SP) ← PC PC ← (2010H)	—	—	—	—	—	—	9

NOTES:

1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
5. Offset is a 2's complement number.
6. Specified bit is one of the 2048 bits in the register file.
7. The "L" (Long) suffix indicates double-word operation.
8. Initiates a Reset by pulling RESET low. Software should re-initialize all the necessary registers with code starting at 2080H.
9. The assembler will not accept this mnemonic.

Mnemonic	Operands	Direct			Immediate			Indirect [Ⓢ]					Indexed [Ⓢ]				
		Opcode	Bytes	State Times	Opcode	Bytes	State Times	Normal		Auto-Inc.		Short		Long			
								Opcode	Bytes	State [Ⓢ] Times	Bytes	State [Ⓢ] Times	Opcode	Bytes	State [Ⓢ] Times	Bytes	State [Ⓢ] Times
Arithmetic Instructions																	
ADD	2	64	3	4	65	4	5	66	3	6/11	3	7/12	67	4	6/11	5	7/12
ADD	3	44	4	5	45	5	6	46	4	7/12	4	8/13	47	5	7/12	6	8/13
ADDB	2	74	3	4	75	3	4	76	3	6/11	3	7/12	77	4	6/11	5	7/12
ADDB	3	54	4	5	55	4	5	56	4	7/12	4	8/13	57	5	7/12	6	8/13
ADDC	2	A4	3	4	A5	4	5	A6	3	6/11	3	7/12	A7	4	6/11	5	7/12
ADDCB	2	B4	3	4	B5	3	4	B6	3	6/11	3	7/12	B7	4	6/11	5	7/12
SUB	2	68	3	4	69	4	5	6A	3	6/11	3	7/12	6B	4	6/11	5	7/12
SUB	3	48	4	5	49	5	6	4A	4	7/12	4	8/13	4B	5	7/12	6	8/13
SUBB	2	78	3	4	79	3	4	7A	3	6/11	3	7/12	7B	4	6/11	5	7/12
SUBB	3	58	4	5	59	4	5	5A	4	7/12	4	8/13	5B	5	7/12	6	8/13
SUBC	2	A8	3	4	A9	4	5	AA	3	6/11	3	7/12	AB	4	6/11	5	7/12
SUBCB	2	B8	3	4	B9	3	4	BA	3	6/11	3	7/12	BB	4	6/11	5	7/12
CMP	2	88	3	4	89	4	5	8A	3	6/11	3	7/12	8B	4	6/11	5	7/12
CMPB	2	98	3	4	99	3	4	9A	3	6/11	3	7/12	9B	4	6/11	5	7/12
MULU	2	6C	3	25	6D	4	26	6E	3	27/32	3	28/33	6F	4	27/32	5	28/33
MULU	3	4C	4	26	4D	5	27	4E	4	28/33	4	29/34	4F	5	28/33	6	29/34
MULUB	2	7C	3	17	7D	3	17	7E	3	19/24	3	20/25	7F	4	19/24	5	20/25
MULUB	3	5C	4	18	5D	4	18	5E	4	20/25	4	21/26	5F	5	20/25	6	21/26
MUL	2	Ⓢ	4	29	Ⓢ	5	30	Ⓢ	4	31/36	4	32/37	Ⓢ	5	31/36	6	32/37
MUL	3	Ⓢ	5	30	Ⓢ	6	31	Ⓢ	5	32/37	5	33/38	Ⓢ	6	32/37	7	33/38
MULB	2	Ⓢ	4	21	Ⓢ	4	21	Ⓢ	4	23/28	4	24/29	Ⓢ	5	23/28	6	24/29
MULB	3	Ⓢ	5	22	Ⓢ	5	22	Ⓢ	5	24/29	5	25/30	Ⓢ	6	24/29	7	25/30
DIVU	2	8C	3	25	8D	4	26	8E	3	28/32	3	29/33	8F	4	28/32	5	29/33
DIVUB	2	9C	3	17	9D	3	17	9E	3	20/24	3	21/25	9F	4	20/24	5	21/25
DIV	2	Ⓢ	4	29	Ⓢ	5	30	Ⓢ	4	32/36	4	33/37	Ⓢ	5	32/36	6	33/37
DIVB	2	Ⓢ	4	21	Ⓢ	4	21	Ⓢ	4	24/28	4	25/29	Ⓢ	5	24/28	6	25/29

270280-27

NOTES:

* Long indexed and Indirect + instructions have identical opcodes with Short indexed and Indirect modes, respectively. The second byte of instructions using any indirect or indexed addressing mode specifies the exact mode used. If the second byte is even, use Indirect or Short Indexed. If it is odd, use Indirect + or Long indexed. In all cases the second byte of the instruction always specifies an even (word) location for the address referenced.

1. Number of state times shown for internal/external operands.

2. The opcodes of signed multiply and divide are the opcodes for the unsigned functions with an "FE" appended as a prefix.

MNEMONIC	OPERANDS	DIRECT			IMMEDIATE			NORMAL			AUTO-INC.		SHORT			LONG	
		OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE TIMES	OPCODE	BYTES	STATE① TIMES	BYTES	STATE① TIMES	OPCODE	BYTES	STATE① TIMES	BYTES	STATE① TIMES
LOGICAL INSTRUCTIONS																	
AND	2	60	3	4	61	4	5	62	3	6/11	3	7/12	63	4	6/11	5	7/12
AND	3	40	4	5	41	5	6	42	4	7/12	4	8/13	43	5	7/12	6	8/13
ANDB	2	70	3	4	71	3	4	72	3	6/11	3	7/12	73	4	6/11	5	7/12
ANDB	3	50	4	5	51	4	5	52	4	7/12	4	8/13	53	5	7/12	6	8/13
OR	2	80	3	4	81	4	5	82	3	6/11	3	7/12	83	4	6/11	5	7/12
ORB	2	90	3	4	91	3	4	92	3	6/11	3	7/12	93	4	6/11	5	7/12
XOR	2	84	3	4	85	4	5	86	3	6/11	3	7/12	87	4	6/11	5	7/12
XORB	2	94	3	4	95	3	4	96	3	6/11	3	7/12	97	4	6/11	5	7/12
DATA TRANSFER INSTRUCTIONS																	
LD	2	A0	3	4	A1	4	5	A2	3	6/11	3	7/12	A3	4	6/11	5	7/12
LDB	2	B0	3	4	B1	3	4	B2	3	6/11	3	7/12	B3	4	6/11	5	7/12
ST	2	C0	3	4	—	—	—	C2	3	7/11	3	8/12	C3	4	7/11	5	8/12
STB	2	C4	3	4	—	—	—	C6	3	7/11	3	8/12	C7	4	7/11	5	8/12
LDBSE	2	BC	3	4	BD	3	4	BE	3	6/11	3	7/12	BF	4	6/11	5	7/12
LDBZE	2	AC	3	4	AD	3	4	AE	3	6/11	3	7/12	AF	4	6/11	5	7/12
STACK OPERATIONS (Internal stack)																	
PUSH	1	C8	2	8	C9	3	8	CA	2	11/15	2	12/16	CB	3	11/15	4	12/16
POP	1	CC	2	12	—	—	—	CE	2	14/18	2	14/18	CF	3	14/18	4	14/18
PUSHF	0	F2	1	8													
POPF	0	F3	1	9													
STACK OPERATIONS (external stack)																	
PUSH	1	C8	2	12	C9	3	12	CA	2	15/19	2	16/20	CB	3	15/19	4	16/20
POP	1	CC	2	14	—	—	—	CE	2	16/20	2	16/20	CF	3	16/20	4	16/20
PUSHF	0	F2	1	12													
POPF	0	F3	1	13													
JUMPS AND CALLS																	
MNEMONIC	OPCODE	BYTES		STATES		MNEMONIC	OPCODE	BYTES		STATES							
LJMP	E7	3		8		LCALL	EF	3		13/16⑤							
SJMP	20-27④	2		8		SCALL	28-2F④	2		13/16⑤							
BR[]	E3	2		8		RET	F0	1		12/16⑤							
						TRAP③	F7	1		21/24							

270280-26

NOTES:

1. Number of state times shown for internal/external operands.
3. The assembler does not accept this mnemonic.
4. The least significant 3 bits of the opcode are concatenated with the following 8 bits to form an 11-bit, 2's complement, offset for the relative call or jump.
5. State times for stack located internal/external.

Conditional Jumps

All conditional jumps are 2 byte instructions. They require 8 state times if the jump is taken, 4 if it is not.

Mnemonic	Opcode	Mnemonic	Opcode	Mnemonic	Opcode	Mnemonic	Opcode
JC	DB	JE	DF	JGE	D6	JGT	D2
JNC	D3	JNE	D7	JLT	DE	JLE	DA
JH	D9	JV	DD	JVT	DC	JST	D8
JNH	D1	JNV	D5	JNVT	D4	JNST	D0

Jump on Bit Clear or Bit Set

These instructions are 3-byte instructions. They require 9 state times if the jump is taken, 5 if it is not.

Mnemonic	Bit Number							
	0	1	2	3	4	5	6	7
JBC	30	31	32	33	34	35	36	37
JBS	38	39	3A	3B	3C	3D	3E	3F

LOOP CONTROL

DJNZ	OPCODE EO;	3 BYTES;	5/9 STATE TIMES (NOT TAKEN/TAKEN)
------	------------	----------	-----------------------------------

Single Register Instructions

Mnemonic	Opcode	Bytes	States	Mnemonic	Opcode	Bytes	States
DEC	05	2	4	EXT	06	2	4
DECB	15	2	4	EXTB	16	2	4
NEG	03	2	4	NOT	02	2	4
NEGB	13	2	4	NOTB	12	2	4
INC	07	2	4	CLR	01	2	4
INCB	17	2	4	CLRB	11	2	4

Shift Instructions

Instr Mnemonic	Word		Instr Mnemonic	Byte		Instr Mnemonic	DBL WD		State Times
	OP	B		OP	B		OP	B	
SHL	09	3	SHLB	19	3	SHLL	0D	3	7 + 1 PER SHIFT(7)
SHR	08	3	SHRB	18	3	SHRL	0C	3	7 + 1 PER SHIFT(7)
SHRA	0A	3	SHRAB	1A	3	SHRAL	0E	3	7 + 1 PER SHIFT(7)

Special Control Instructions

Mnemonic	Opcode	Bytes	States	Mnemonic	Opcode	Bytes	States
SETC	F9	1	4	DI	FA	1	4
CLRC	F8	1	4	EI	FB	1	4
CLRV	FC	1	4	NOP	FD	1	4
RST (6)	FF	1	166	SKIP	00	2	4

Normalize

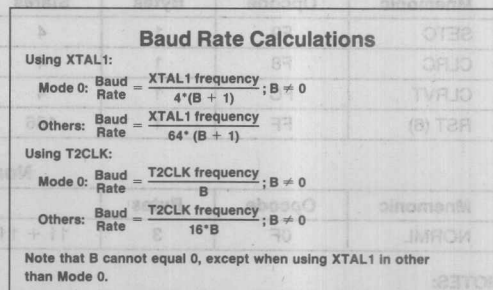
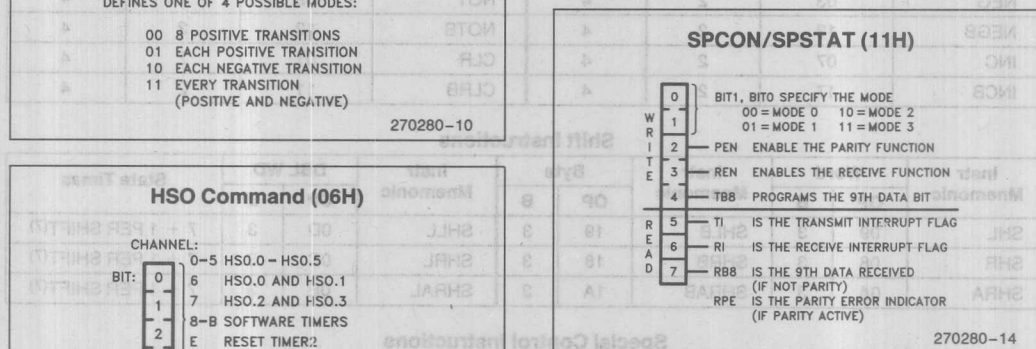
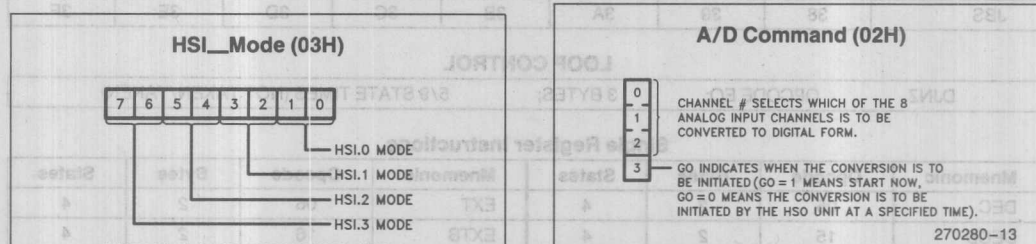
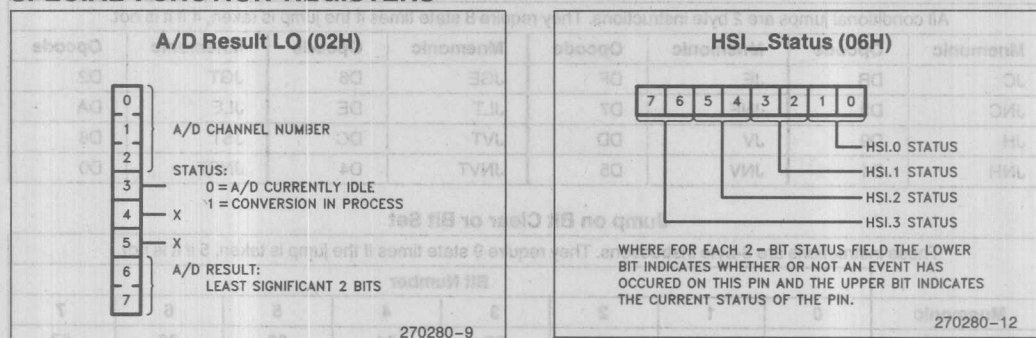
Mnemonic	Opcode	Bytes	State Times
NORML	0F	3	11 + 1 PER SHIFT

NOTES:

6. This instruction takes 2 states to pull RST low, then holds it low for 2 states to initiate a reset. The reset takes 12 states, at which time the program restarts at location 2080H.

7. Execution will take at least 8 states, even for 0 shift.

SPECIAL FUNCTION REGISTERS



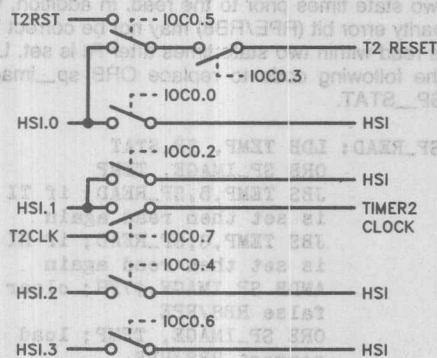
SPECIAL FUNCTION REGISTERS (Continued)

IOC0 (15H)

- 0 — HSI.0 INPUT ENABLE / DISABLE
- 1 — TIMER 2 RESET EACH WRITE
- 2 — HSI.1 INPUT ENABLE / DISABLE
- 3 — TIMER 2 EXTERNAL RESET ENABLE / DISABLE
- 4 — HSI.2 INPUT ENABLE / DISABLE
- 5 — TIMER 2 RESET SOURCE HSI.0 / T2RST
- 6 — HSI.3 INPUT ENABLE / DISABLE
- 7 — TIMER 2 CLOCK SOURCE HSI.1 / T2CLK

270280-15

IOC0 (15H)



270280-16

IOS0 (15H)

- 0 — HSO.0 CURRENT STATE
- 1 — HSO.1 CURRENT STATE
- 2 — HSO.2 CURRENT STATE
- 3 — HSO.3 CURRENT STATE
- 4 — HSO.4 CURRENT STATE
- 5 — HSO.5 CURRENT STATE
- 6 — CAM OR HOLDING REGISTER IS FULL
- 7 — HSO HOLDING REGISTER IS FULL

270280-17

IOC1 (16H)

- 0 — SELECT PWM / SELECT P2.5
- 1 — EXTERNAL INTERRUPT ACH7 / EXTINT
- 2 — TIMER 1 OVERFLOW INTERRUPT ENABLE / DISABLE
- 3 — TIMER 2 OVERFLOW INTERRUPT ENABLE / DISABLE
- 4 — HSO.4 OUTPUT ENABLE / DISABLE
- 5 — SELECT TXD / SELECT P2.0
- 6 — HSO.5 OUTPUT ENABLE / DISABLE
- 7 — HSI INTERRUPT
FIFO FULL / HOLDING REGISTER LOADED

270280-18

Vector	Vector Location		Priority
	(High Byte)	(Low Byte)	
Software	2011H	2010H	Not Applicable
Extint	200FH	200EH	
Serial Port	200DH	200CH	6
Software	200BH	200AH	5
Timers			
HSI.0	2009H	2008H	4
High Speed	2007H	2006H	3
Outputs			
HSI Data	2005H	2004H	2
Available			
A/D Conversion	2003H	2002H	1
Complete			
Timer Overflow	2001H	2000H	0 (Lowest)

IOS1 (16H)

- 0 — SOFTWARE TIMER 0 EXPIRED
- 1 — SOFTWARE TIMER 1 EXPIRED
- 2 — SOFTWARE TIMER 2 EXPIRED
- 3 — SOFTWARE TIMER 3 EXPIRED
- 4 — TIMER 2 HAS OVERFLOW
- 5 — TIMER 1 HAS OVERFLOW
- 6 — HSI FIFO IS FULL
- 7 — HSI HOLDING REGISTER DATA AVAILABLE

270280-19

FUNCTIONAL DEVIATIONS

Functional deviations from the 809X and 839X on the 809X-90 and 839X-90.

CPU Section

1. Indexed, 3 Operand Multiply—The displacement portion of an indexed, three word multiply may not be in the range of 200H thru 17FFH inclusive. This also includes byte multiples that use 3-operands.
2. Add or Subtract with carry—The zero flag is both set and cleared by these instructions. Zero checking must be done after each operation.
3. EXT—This instruction never sets the N flag, and always sets the Z flag. The EXTB works correctly. Check the flags before executing an EXT instruction. Additionally, having more than 2 wait states during EXT (extend word only) instruction may cause the instruction to produce incorrect results.
4. Read-Modify-Write on Interrupt Pending—A read-modify-write instruction on the interrupt pending register may cause interrupts that occur during execution of the instruction to be missed.
5. READY line—The READY line should not be brought low during the execution of an instruction that accesses HSI_TIME, SP_STAT or IOS1. It should also not be brought low for a data write during the instruction immediately preceding one of the above operations. Do not use wait states for program memory that holds these instructions. Also place an NOP between writes to slow memory and accesses to HSO_TIME, SP_STAT, or IOS1.
The READY line also should not be brought low for more than two state times when using the EXT (extend word) instruction.
6. Signed Divide—The V and VT flags may indicate an overflow after a signed divide when no overflow has occurred.
7. The sticky flag is not affected when a shift by zero is executed on an 8X9X-90.
8. The JBS and JBC instructions should not be used directly on P2.1 or any pins of Port 0, if used as digital I/O. If it is necessary to test these bits, first load the port data into a temporary register, and then test the bit there.
9. The first few instructions of an interrupt service routine should check IOS1.7 and exit if the Hold-Register is not loaded. This will successfully clear all unwanted events.

HSI/HSO Section

1. HSI Timing—An event occurring within 16 state times of a prior event on the same HSI line may

not be recorded. Additionally, an event occurring within 16 state times of a prior event on another HSI line may be recorded with a time tag one count earlier than expected. Events are defined as the condition the line is set to trigger on. The effective resolution is increased to 4 μ s for such closely spaced events.

2. HSI Divide by 8 Mode—If an event on a pin set to look for every eighth transition occurs less than 16 state times after an event on any other pin, then the divide by 8 event will be recorded twice in the HSI FIFO. The time tag of the duplicate FIFO entry will be equal to that of the initial entry plus one. The programmer's software should detect and discard the second entry.
3. HSO Interrupts—Software timer interrupts cannot be generated by the HSO commands that reset Timer 2 or start an A to D conversion.

Serial Port Section

1. Serial Port Flags—Reading SP_STAT may not clear the TI or RI flag if that flag was set within two state times prior to the read. In addition, the parity error bit (RPE/RB8) may not be correct if it is read within two state times after RI is set. Use the following code to replace ORB sp_image, SP_STAT.

```
SP_READ: LDB TEMP, SP_STAT
          ORB SP_IMAGE, TEMP
          JBS TEMP,5,SP_READ; if TI
          is set then read again
          JBS TEMP,6,SP_READ; if RI
          is set then read again
          ANDB SP_IMAGE,#7FH; clear
          false RB8/RPE
          ORB SP_IMAGE, TEMP; load
          correct RB8/RPE
```

2. Serial Port Mode 0—The serial port is not tested in Mode 0. The receive function in this mode does not work correctly. The receive function will not work unless the first bit shifted in is a one.
3. Serial Port Baud Value—Loading the baud rate register with 8000H (maximum baud rate, internal clock) may cause an 11 millisecond delay (at Fosc = 12 MHz) before the port is properly initialized. After initialization the port works properly.

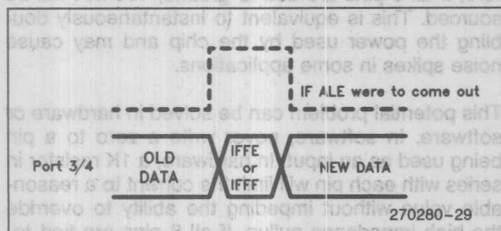
Include a 44000 state time delay after writing 8000H to the Baud Rate Register.

Standard I/O Section

1. Ports 3 and 4 (Internal Execution Mode Only)—To be used as outputs, Ports 3 and 4 each must be addressed as words but written to as bytes. To write to Port 3 use "ST temp, 1ffeh", where the low byte of "temp" contains the data for the port. To write to Port 4, use the DCB operator to gener-

ate the opcode sequence "0C3H, 001H, 0FFH, 01FH, (temp)", where the high byte of "temp" contains the data for the port. Ports 3 and 4 will not work as input ports.

Additionally, when using Ports 3 and 4 as outputs, the address IFFE_H or IFFF_H (depending on a write to Port 3 or 4) will appear on the bus. No control signals are present, just bus data.



ADDITIONAL INFORMATION

The following information was not in the 1984 "8096 Users Manual"

1. After a chip reset the Watchdog Timer will not run until a "01EH" followed by a "0E1H" is written to the Watchdog Timer register. After this is done the Watchdog Timer functions as described in the users manual until the next chip reset. This feature permits disabling of the Watchdog by simply not writing to it.

2. External interrupts on P0.7 are sampled every state time instead of every eight state times.
3. The baud rate generated in the external clock mode is four times faster than stated in the Users Manual. The correct formula for other than mode 0 using T2CLK as the input frequency is:
Baud Rate = Input Frequency/(16*B).
4. If more than one HSO event is scheduled to occur with interrupts at the same time multiple HSO interrupts may occur. This is because HSO interrupts are internal events and as such are not synchronized to Timer 1.
5. Locations 2012H through 207FH in external memory must be filled with the hex value 0FFFFH to ensure compatibility with future parts. The internal locations in this range are still reserved for the factory test code.
6. There are several restrictions on using the special function registers.

- A. Neither the source nor the destination addresses of the Multiply and Divide instructions can be a writable special function register.
- B. These registers may not be used as base or index registers for indexed or indirect instructions.
- C. Several of these registers can only be accessed as words, while others only as bytes. These restrictions are listed in Chapter 3 of the manual.

MCS®-96 Thermal Characteristics

Package	Theta jA	Theta jC
48 Ld Ceramic DIP	26°C/W	6.5°C/W
68 Ld Ceramic Grid Array	35°C/W	10°C/W
68 Ld Plastic Leaded Chip Carrier	37°C/W	10°C/W
68 Ld Plastic Flat Pack	36°C/W	12°C/W

Theta jA = Thermal resistance between junction and the surrounding environment (ambient). Measurement taken 1 ft. away from case in air flow environment.

Theta jC = Thermal resistance between junction and package surface (case).

All values of theta jA and theta jC may fluctuate depending on the environment (with or without air flow, and how much air flow) and device I_{CC} and V_{CC} of operation. Typical variations are ±2°C/Watt.

DRIVE AND INTERFACE LEVELS

There are 5 types of I/O lines on the 8096. Of these, 2 are inputs and 3 are outputs. All of the pins of the same type have the same current/voltage characteristics. Some of the input pins, such as XTAL1 and RESET, may have slightly different characteristics.

While discussing the characteristics of the I/O pins some approximate current and voltage specifications will be given. The exact specifications are available in the DC section of this data sheet.

Quasi-Bidirectional Ports

The quasi-bidirectional port is both an input and an output port. It has three states, low impedance current sink, low impedance current source, and high impedance current source. As a low impedance current sink, the pin has specification of sinking up to around 0.4 milliamps, while staying below the 0.45 volt level. The pin is placed in this condition by writing a '0' the SFR (Special Function Register) controlling the pin.

When a '1' is written to the SFFI location controlling the pin, a low impedance current source is turned on for one state time, then it is turned off and the depletion pullup holds the line at a logical '1' state. The low impedance pullup is used to shorten the rise time of the pin, and has current source capability on the order of 100 times that of the depletion pullup. The configuration of a quasi-bidirectional port pin is shown in Figure 9.

While the depletion mode pullup is the only device on, the pin may be used as an input with a leakage of around 100 microamps from 0.45 volts to V_{CC} . It is ideal for use with TTL and CMOS chips and may even be used directly with switches, however if the switch option is used certain precautions should be taken. It is important to note that any time the pin is read, the value returned will be the value on the pin, not the value placed in the control register. This could prevent logical operations on these pins while they are used as inputs.

Quasi-Bidirectional Hardware Connections

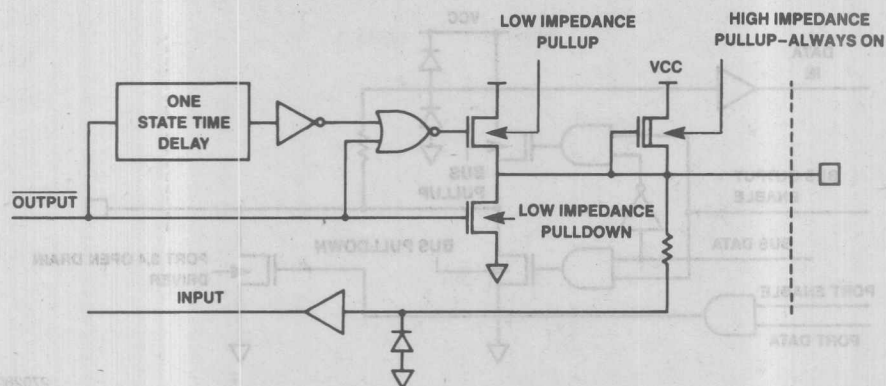
When using the quasi-bidirectional ports as inputs tied to switches, series resistance may be needed if the ports will be written to internally after the part is initialized. The amount of current sourced to ground by each pin is typically 20 milliamps or more. Therefore, if all 8 pins are tied to ground, 160 mA will be sourced. This is equivalent to instantaneously doubling the power used by the chip and may cause noise spikes in some applications.

This potential problem can be solved in hardware or software. In software, never write a zero to a pin being used as an input. In hardware, a 1K resistor in series with each pin will limit the current to a reasonable value without impeding the ability to override the high impedance pullup. If all 8 pins are tied together, a 120 Ω resistor would be reasonable. The problem is not quite as severe when the inputs are tied to electronic devices instead of switches, as most external pulldowns will not hold 20 mA to 0.0 volts.

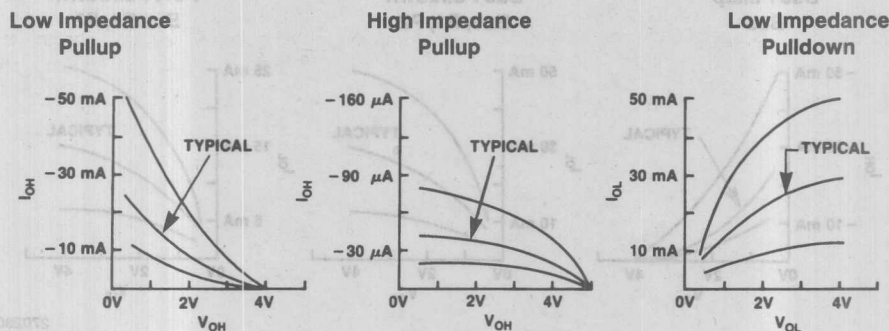
If a switch is used on a long line connected to a quasi-bidirectional pin, a pullup resistor is recommended to reduce the possibility of noise glitches and to decrease the rise time of the line. On extremely long lines that are handling slow signals a capacitor may be helpful in addition to the resistor to reduce noise.

Input Ports, Analog and Digital

The high impedance input ports on the 8096 have an input leakage of a few microamps and are predominantly capacitive loads on the order of 10 pF. The Port 0 pins have an additional function when the A/D converter is being used. These pins are the input to the A/D converter, and as such, are required to provide current to the comparator when the conversion is in process. This means that the input characteristics of a pin will change if a conversion is being done on that pin. For further information see the MCS-96 Architectural Overview Chapter of the Embedded Controller Handbook.



270280-20



270280-21

NOTE:

These graphs show typical pin capabilities, they are not guaranteed specifications

Figure 9. Quasi-Bidirectional Port

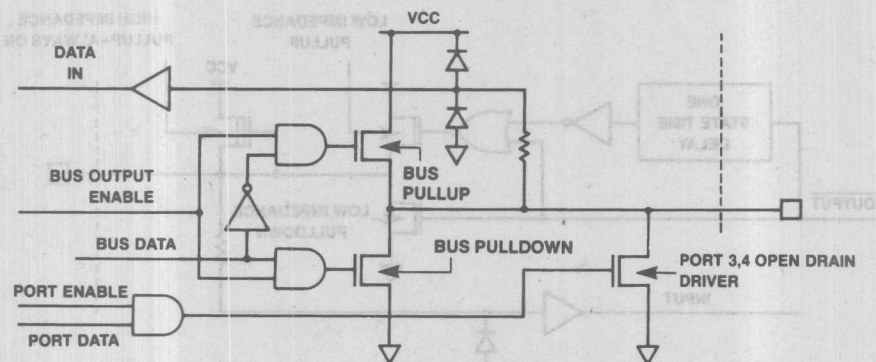
Open Drain Ports

Ports 3 and 4 on the 8096 are open drain ports. There is no pullup when these pins are used as I/O ports. These pins have different characteristics when used as bus pins. A diagram of the output buffers connected to Ports 3 and 4 and the bus pins is shown in Figure 10.

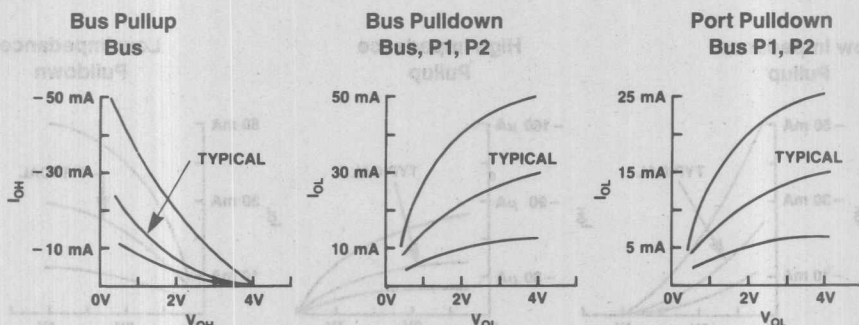
When the Ports 3 and 4 are used as inputs, or as bus pins, they must first be written with a '1', this will put the ports in a high impedance mode. When they are used as outputs, a pullup resistor **MUST** be used externally. The sink capability of these pins is on the order of 0.4 milliamps so the total pullup current to the pin must be less than this. A 15K pullup resistor will source a maximum of 0.33 milliamps, so it would be a reasonable value to choose if no other circuits with pullups were connected to the pin.

HSO Pins, Control Outputs and Bus Pins

The control outputs and HSO pins have output buffers with the same output characteristics as those of the bus pins. Included in this category of control outputs are: TXD, RXD (in Mode 0), PWM, CLKOUT, ALE, BHE, RD, and WR. The bus pins have three states: output high, output low, and high impedance input. As a high output, the pins are specified to source around 200 μ A to 2.4 volts, but the pins can source on the order of ten times that value in order to provide fast rise times. When used as a low output, the pins can sink around 2 mA at 0.45V, and considerably more as the voltage increases. When in the high impedance state, the pins act as a capacitive load with a few microamps of leakage. Figure 10 gives the internal configuration of a bus pin and its typical pin capabilities.



270280-22



270280-23

NOTE:

These graphs show typical pin capabilities, they are not guaranteed specifications

Figure 10. Bus and Port 3 and 4 Pins

The control outputs and H2O pins have output buffers with the same output characteristics as those of the bus pins. Included in this category of control outputs are: TXD, RXD, RxD in Mode 0, RxD in Mode 1, ALE, BHC, RD, and WR. The bus pins have three states: output high, output low, and high impedance. As a high output, the pins are specified to source around 500 μ A to 5.4 V, but the pins can source on the order of ten times that value in order to provide fast rise time. When used as a low output, the pins can sink around 5 mA at 0.45 V, and considerably more as the voltage increases. When the high impedance state, the pins act as a capacitor loaded with a few picofarads of leakage. Figure 10 gives the internal configuration of a bus pin and its typical pin capabilities.

Ports 3 and 4 on the 8088 are open drain ports. There is no pullup when these pins are used as I/O ports. These pins have different characteristics when used as bus pins. A diagram of the output buffer is shown in Figure 10.

When the Ports 3 and 4 are used as inputs, or as bus pins, they must first be written with a "1". This will put the pins in a high impedance mode. When they are used as outputs, a pullup resistor MUST be used externally. The sink capability of these pins is on the order of 0.4 milliamperes so the total pullup current to the pin must be less than this. A 18K pullup resistor will source a maximum of 0.33 milliamperes, so it would be a reasonable value to choose if no other circuits with pullups were connected to the pin.

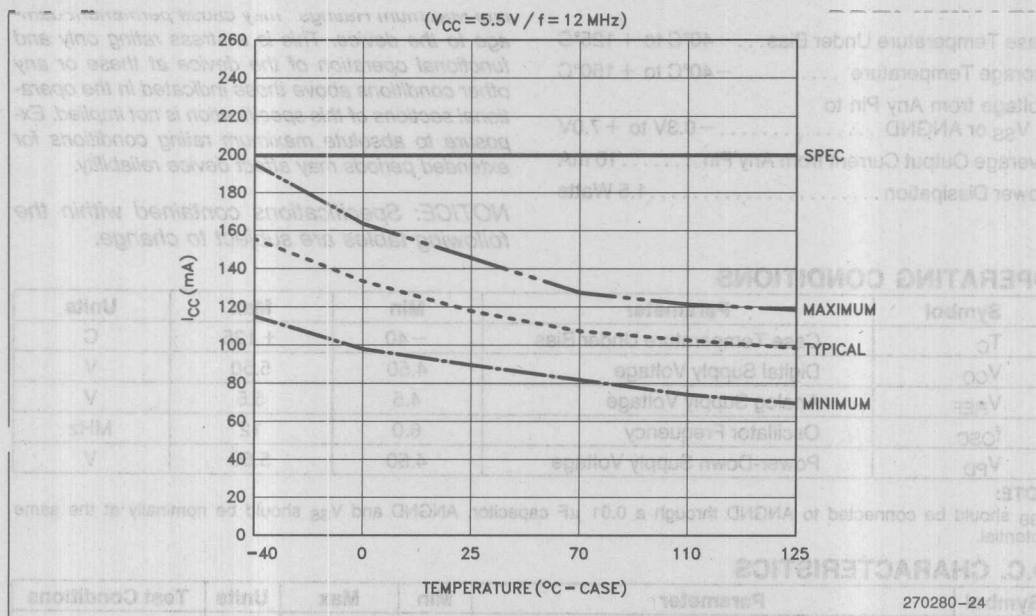


Figure 11. 809X-90, 839X-90 I_{CC} vs. Temp

Figure 11 is an I_{CC} vs. temperature for the MCS-96 family. Test conditions are stated and actuals may vary depending on speed, package, V_{CC} and temperature, but will not exceed maximum limit of 200 mA at any case temperature between -40°C through $+125^{\circ}\text{C}$.

I_{CC}	V_{CC} Supply Current	200	mA	All Outputs Disconnected
I_{PD}	VPD Supply Current	1	mA	Normal operation and Power-Down
I_{REF}	V_{REF} Supply Current	8	mA	
I_{I1}	Input Leakage Current to all pins of HSI, P2, P4 and P5.1	± 10	μA	$V_{IN} = 0$ to V_{CC}
I_{I2}	Input Leakage to Port 0	± 3	μA	$V_{IN} = 0$ to V_{CC}
I_{I3}	Input High Current to EX	100	μA	$V_{IH} = 2.5V$
I_{I4}	Input Low Current to all pins of P1 and P2.8, P2.7	-100	μA	$V_{IL} = 0.45V$
I_{I5}	Input Low Current to RESET	-5	μA	$V_{IL} = 0.45V$
I_{I6}	Input Low Current P2.5, P2.6, P2.4, READY	-10	μA	$V_{IL} = 0.45V$
C_d	Pin Capacitance (Any Pin to V_{CC})	10	pF	TEST = 1.0 MHz

NOTES:
1. I_{OL} = 0.50 mA for all pins of P1, P2.8 and P2.7, and for all pins of P2 and P4 when used as ports; I_{OL} = 5.0 mA for TXD, RXD (in serial port mode), PWM, CLKOUT, ALE, BHE, RD, WR, and RESET, and all pins of HSD and P3 and P4 when used as external memory bus (AD0-AD15).
2. I_{OH} = -20 μA for all pins of P1, P2.8 and P2.7, I_{OH} = -500 μA for TXD, RXD (in serial port mode), PWM, CLKOUT, ALE, BHE, RD, WR, and all pins of HSD and P3 and P4 when used as external memory bus (AD0-AD15), P3 and P4 when used as ports, have open-drain outputs.
3. The values are not tested in production and are based on theoretical estimates and laboratory tests.

ABSOLUTE MAXIMUM RATINGS*

Case Temperature Under Bias	−40°C to +125°C
Storage Temperature	−40°C to +150°C
Voltage from Any Pin to V _{SS} or ANGND	−0.3V to +7.0V
Average Output Current from Any Pin	10 mA
Power Dissipation	1.5 Watts

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
T _C	Case Temperature Under Bias	−40	+125	C
V _{CC}	Digital Supply Voltage	4.50	5.50	V
V _{REF}	Analog Supply Voltage	4.5	5.5	V
f _{OSC}	Oscillator Frequency	6.0	12	MHz
V _{PD}	Power-Down Supply Voltage	4.50	5.50	V

NOTE:

V_{BB} should be connected to ANGND through a 0.01 μF capacitor. ANGND and V_{SS} should be nominally at the same potential.

D.C. CHARACTERISTICS

Symbol	Parameter	Min	Max	Units	Test Conditions
V _{IL}	Input Low Voltage (Except RESET)	−0.3	+0.8	V	
V _{IL1}	Input Low Voltage, RESET	−0.3	+0.7	V	
V _{IH}	Input High Voltage (Except RESET, NMI, XTAL1)	2.0	V _{CC} + 0.5	V	
V _{IH1}	Input High Voltage, RESET Rising	2.4	V _{CC} + 0.5	V	
V _{IH2}	Input High Voltage, RESET Falling	2.1	V _{CC} + 0.5	V	
V _{IH3}	Input High Voltage, NMI, XTAL1	2.4	V _{CC} + 0.5	V	
V _{OL}	Output Low Voltage ⁽¹⁾		0.5	V	
V _{OH}	Output High Voltage ⁽²⁾	2.4		V	
I _{CC}	V _{CC} Supply Current		200	mA	All Outputs Disconnected
I _{PD}	VPD Supply Current		1	mA	Normal operation and Power-Down
I _{REF}	V _{REF} Supply Current		8	mA	
I _{LI}	Input Leakage Current to all pins of HSI, P3, P4, and to P2.1		±10	μA	V _{in} = 0 to V _{CC}
I _{LI1}	Input Leakage to Port 0		±3	μA	V _{IN} = 0 to V _{CC}
I _{IH}	Input High Current to EA		100	μA	V _{IH} = 2.4V
I _{IL}	Input Low Current to all pins of P1, and to P2.6, P2.7		−100	μA	V _{IL} = 0.45V
I _{IL1}	Input Low Current to RESET	0.3	−2	mA	V _{IL} = 0.45V
I _{IL2}	Input Low Current P2.2, P2.3, P2.4, READY		−50	μA	V _{IL} = 0.45V
C _S	Pin Capacitance (Any Pin to V _{SS}) ⁽³⁾		10	pF	f _{TEST} = 1.0 MHz

NOTES:

1. I_{OL} = 0.36 mA for all pins of P1, for P2.6 and P2.7, and for all pins of P3 and P4 when used as ports. I_{OL} = 2.0 mA for TXD, RXD (in serial port mode 0), PWM, CLKOUT, ALE, BHE, RD, WR, and RESET and all pins of HSO and P3 and P4 when used as external memory bus (AD0–AD15).

2. I_{OH} = −20 μA for all pins of P1, for P2.6 and P2.7. I_{OH} = −200 μA for TXD, RXD (in serial port mode 0), PWM, CLKOUT, ALE, BHE, WR, and all pins of HSO and P3 and P4 when used as external memory bus (AD0–AD15). P3 and P4, when used as ports, have open-drain outputs.

3. The values are not tested in production and are based on theoretical estimates and laboratory tests.

A/D CONVERTER SPECIFICATIONS

The absolute conversion accuracy is dependent on the accuracy of V_{REF} . The specifications given below assume adherence to the Operating Conditions section of these data sheets. Testing is done at $V_{REF} = 5.120V$.

OPERATING CONDITIONS

V_{CC}, V_{PD}, V_{REF} 4.5V to 5.5V
 $V_{SS}, ANGND$ 0.0V
 T_C -40°C to +125°C
 F_{OSC} 6.0 to 12.0 MHz

Test Conditions:
 $V_{REF} = 5.120V$

Parameter	Typical*(1)	Minimum	Maximum	Units**	Notes
Resolution		1024 10	1024 10	Levels Bits	
Absolute Error		0	±4	LSBs	
Full Scale Error	-0.5 ± 0.5			LSBs	
Zero Offset Error	±0.5			LSBs	
Non-Linearity		0	±4	LSBs	
Differential Non-Linearity		0	±2	LSBs	
Channel-to-Channel Matching		0	±1	LSBs	
Repeatability	±0.25			LSBs	1
Temperature Coefficients:					
Offset	0.009			LSB/°C	1
Full Scale	0.009			LSB/°C	1
Differential Non-Linearity	0.009			LSB/°C	1
Off Isolation		-60		dB	1, 2, 3
Feedthrough	-60			dB	1, 2
V_{CC} Power Supply Rejection	-60			dB	1, 2
Input Resistance		1K	5K	Ω	1
D.C. Input Leakage		0	3.0	μA	

NOTES:

* These values are expected for most parts at 25°C.

** An "LSB", as used here, is defined in the glossary which follows and has a value of approximately 5 mV.

1. These values are not tested in production and are based on theoretical estimates and laboratory tests.

2. DC to 100 KHz.

3. Multiplexer Break-Before-Make Guaranteed.

A/D GLOSSARY OF TERMS

ABSOLUTE ERROR—The maximum difference between corresponding actual and ideal code transitions. Absolute Error accounts for all deviations of an actual converter from an ideal converter.

ACTUAL CHARACTERISTIC—The characteristic of an actual converter. The characteristic of a given converter may vary over temperature, supply voltage, and frequency conditions. An actual characteristic rarely has ideal first and last transition locations or ideal code widths. It may even vary over multiple conversions under the same conditions.

BREAK-BEFORE-MAKE—The property of a multiplexer which guarantees that a previously selected channel will be deselected before a new channel is selected. (e.g. the converter will not short inputs together.)

CHANNEL-TO-CHANNEL MATCHING—The difference between corresponding code transitions of actual characteristics taken from different channels under the same temperature, voltage and frequency conditions.

CHARACTERISTIC—A graph of input voltage versus the resultant output code for an A/D converter. It describes the transfer function of the A/D converter.

CODE—The digital value output by the converter.

CODE CENTER—The voltage corresponding to the midpoint between two adjacent code transitions.

CODE TRANSITION—The point at which the converter changes from an output code of Q, to a code of Q + 1. The input voltage corresponding to a code transition is defined to be that voltage which is equally likely to produce either of two adjacent codes.

CODE WIDTH—The voltage corresponding to the difference between two adjacent code transitions.

CROSSTALK—See "Off-Isolation".

D.C. INPUT LEAKAGE—Leakage current to ground from an analog input pin.

DIFFERENTIAL NON-LINEARITY—The difference between the ideal and actual code widths of the terminal based characteristic.

FEEDTHROUGH—Attenuation of a voltage applied on the selected channel of the A/D Converter after the sample window closes.

FULL SCALE ERROR—The difference between the expected and actual input voltage corresponding to the full scale code transition.

IDEAL CHARACTERISTIC—A characteristic with its first code transition at $V_{IN} = 0.5 \text{ LSB}$, its last code transition at $V_{IN} = (V_{REF} - 1.5 \text{ LSB})$ and all code widths equal to one LSB.

INPUT RESISTANCE—The effective series resistance from the analog input pin to the sample capacitor.

LSB—Least Significant Bit: The voltage corresponding to the full scale voltage divided by 2^n , where n is the number of bits of resolution of the converter. For a 10-bit converter with a reference voltage of 5.12V, one LSB is 5.0 mV. Note that this is different than digital LSBs, since an uncertainty of two LSB, when referring to an A/D converter, equals 10 mV. (This has been confused with an uncertainty of two digital bits, which would mean four counts, or 20 mV.)

MONOTONIC—The property of successive approximation converters which guarantees that increasing input voltages produce adjacent codes of increasing value, and that decreasing input voltages produce adjacent codes of decreasing value.

NO MISSED CODES—For each and every output code, there exists a unique input voltage range which produces that code only.

NON-LINEARITY—The maximum deviation of code transitions of the actual characteristic from the corresponding code transitions of the terminal based characteristic of the converter.

OFF-ISOLATION—Attenuation of a voltage applied on a deselected channel of the A/D converter. (Also referred to as Crosstalk.)

REPEATABILITY—The difference between corresponding code transitions from different actual characteristics taken from the same converter on the same channel at the same temperature, voltage and frequency conditions.

RESOLUTION—The number of input voltage levels that the converter can unambiguously distinguish between. Also defines the number of useful bits of information which the converter can return.

SUCCESSIVE APPROXIMATION—An A/D conversion method which uses a binary search to arrive at the best digital representation of an analog input.

TEMPERATURE COEFFICIENTS—Change in the stated variable per degree centigrade temperature change. Temperature coefficients are added to the typical values of a specification to see the effect of temperature drift.

TERMINAL BASED CHARACTERISTIC—An actual characteristic which has been rotated and translated to remove zero offset and full scale error.

V_{CC} REJECTION—Attenuation of noise on the V_{CC} line to the A/D converter.

ZERO OFFSET—The difference between the expected and actual input voltage corresponding to the first code transition.

Timing Responses (MCS-86 parts meet these specs)

Symbol	Parameter	Min	Max	Units
FXAL	Oscillator Frequency	8.00	12.00	MHz
Tosc	Oscillator Period	83	108	ns
TOHCH	Oscillator High to CLKOUT High	0	150	ns
TOHCH	CLKOUT Period	3Tosc	3Tosc	ns
TOHCH	CLKOUT High Time	Tosc - 20	Tosc + 50	ns
TOHCH	CLKOUT Low to ALE High	-25	50	ns
TOHCH	ALE Low to CLKOUT High	Tosc - 20	Tosc + 40	ns
TOHCH	ALE Pulse Width	Tosc - 30	Tosc + 50	ns
TAVAL	Address Setup to End of ALE	Tosc - 80		ns
TLRL	End of ALE to RDY or WR Active	Tosc - 20		ns
TLAX	Address Hold After End of ALE	Tosc - 20		ns
TWLVH	WR Pulse Width	2Tosc - 35		ns
TOVWX	Output Data Setup to End of WR	2Tosc - 80		ns
TWVOX	Output Data Hold After End of WR	Tosc - 25		ns
TWVXH	End of WR to Next ALE	2Tosc - 30		ns
TRLRH	RDY Pulse Width	3Tosc - 30		ns
TRLH	End of RDY to Next ALE	Tosc - 25		ns

NOTES:
1. If more than one wait state is desired, add 3Tosc for each additional wait state.
2. This specification is not tested, but is verified by design analysis and/or derived from other tested parameters.
3. CLKOUT is directly generated as a divide by 3 of the oscillator. The period will be 3Tosc ± 10 ns. Tosc is constant and the rise and fall times on XTAL 1 are less than 10 ns. CLKOUT is not loaded out on 40-pin parts.

A.C. CHARACTERISTICS

(VCC, VPD = 4.5 to 5.5 Volts; $T_C = -40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$; fosc = 6.0 to 12.0 MHz)

Test Conditions: Load Capacitance on Output Pins = 80 pF

Oscillator Frequency = 12.00 MHz

TIMING REQUIREMENTS (Other system components must meet these specs.)

Symbol	Parameter	Min	Max	Units
TCLYX	READY Hold after CLKOUT Edge	0		ns
TLlyV	End of ALE to READY Setup	-Tosc	2Tosc-60	ns
TLlyH	End of ALE to READY High	2Tosc+60	4Tosc-60(1)	ns
TYLYH	Non-ready Time		1000	ns
TAVDV	Address Valid to Input Data Valid		5Tosc-100	ns
TRLDV	RD/Active to Input Data Valid		3Tosc-70	ns
TRDX	Data Hold after RD/inactive(2)	0		ns
TRXDZ	RD/Inactive to Input Data Float(2)		Tosc-20	ns

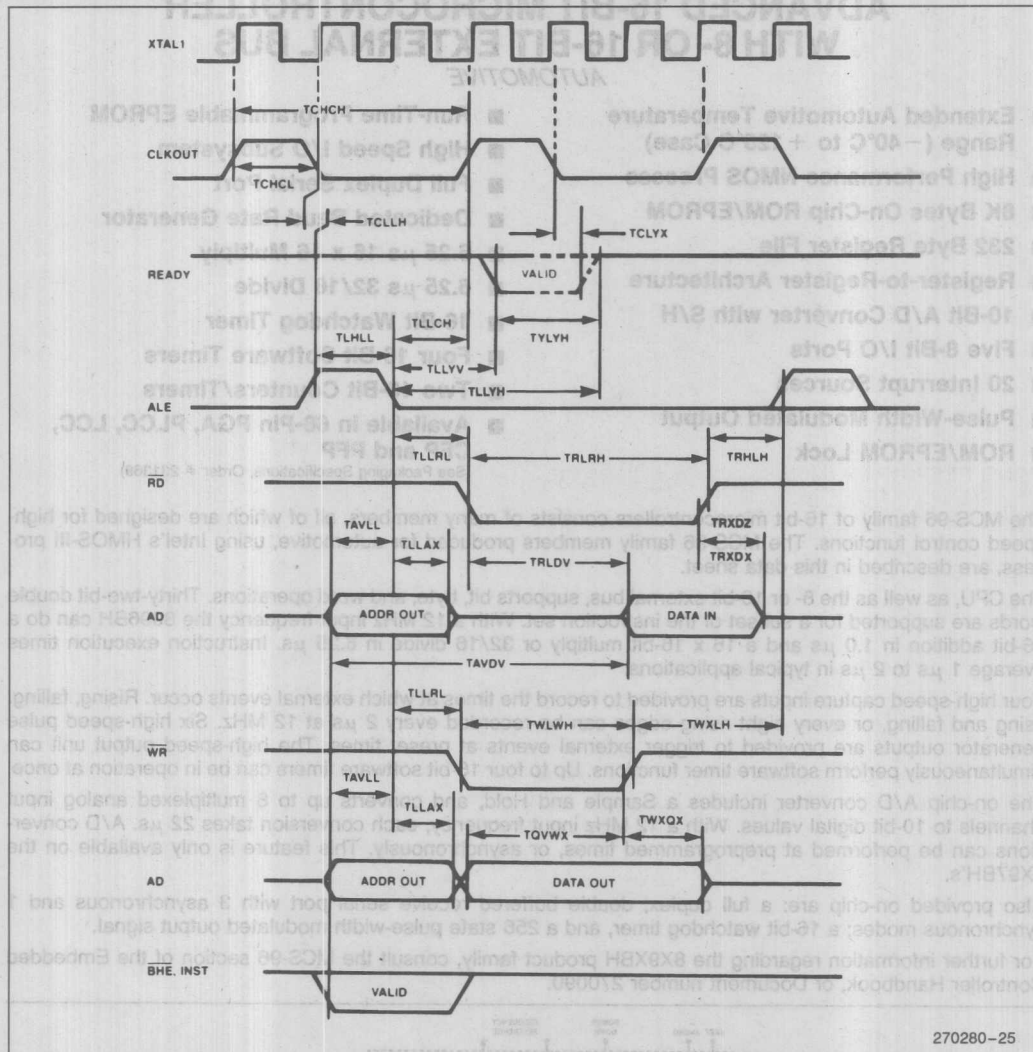
TIMING RESPONSES (MCS-96 parts meet these specs.)

Symbol	Parameter	Min	Max	Units
FXTAL	Oscillator Frequency	6.00	12.00	MHz
Tosc	Oscillator Period	83	166	ns
TOHCH	Oscillator High to CLKOUT High(3)	0	120	ns
TCHCH	CLKOUT Period(2)	3Tosc(3)	3Tosc(3)	ns
TCHCL	CLKOUT High Time	Tosc-20	Tosc+20	ns
TCLLH	CLKOUT Low to ALE High	-25	20	ns
TLLCH	ALE Low to CLKOUT High	Tosc-20	Tosc+40	ns
TLHLL	ALE Pulse Width	Tosc-30	Tosc+20	ns
TAVLL	Address Setup to End of ALE	Tosc-60		ns
TLLRL	End of ALE to RD/ or WR/ Active	Tosc-20		ns
TLLAX	Address Hold After End of ALE	Tosc-20		ns
TWLWH	WR/ Pulse Width	2Tosc-35		ns
TQVWX	Output Data Setup to End of WR/	2Tosc-60		ns
TWXQX	Output Data Hold After End of WR/	Tosc-25		ns
TWXLH	End of WR/ to Next ALE	2Tosc-30		ns
TRLRH	RD/ Pulse Width	3Tosc-30		ns
TRHLH	End of RD/ to Next ALE	Tosc-25		ns

NOTES:

1. If more than one wait state is desired, add 3Tosc for each additional wait state.
2. This specification is not tested, but is verified by design analysis and/or derived from other tested parameters.
3. CLKOUT is directly generated as a divide by 3 of the oscillator. The period will be $3Tosc \pm 10$ ns if Tosc is constant and the rise and fall times on XTAL 1 are less than 10 ns. CLKOUT is not bonded out on 48-pin parts.

WAVEFORM



270280-25

Bus Signal Timings

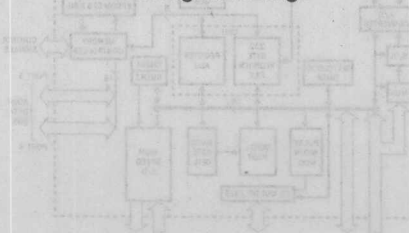


Figure 1. MCS-96 Block Diagram

MCS®-96 809XBH/839XBH/879XBH ADVANCED 16-BIT MICROCONTROLLER WITH 8- OR 16-BIT EXTERNAL BUS AUTOMOTIVE

- Extended Automotive Temperature Range (−40°C to +125°C Case)
- High Performance NMOS Process
- 8K Bytes On-Chip ROM/EPROM
- 232 Byte Register File
- Register-to-Register Architecture
- 10-Bit A/D Converter with S/H
- Five 8-Bit I/O Ports
- 20 Interrupt Sources
- Pulse-Width Modulated Output
- ROM/EPROM Lock
- Run-Time Programmable EPROM
- High Speed I/O Subsystem
- Full Duplex Serial Port
- Dedicated Baud Rate Generator
- 6.25 μ s 16 x 16 Multiply
- 6.25 μ s 32/16 Divide
- 16-Bit Watchdog Timer
- Four 16-Bit Software Timers
- Two 16-Bit Counters/Timers
- Available in 68-Pin PGA, PLCC, LCC, CFP and PFP
(See Packaging Specifications, Order # 231369)

The MCS-96 family of 16-bit microcontrollers consists of many members, all of which are designed for high-speed control functions. The MCS-96 family members produced for automotive, using Intel's HMOS-III process, are described in this data sheet.

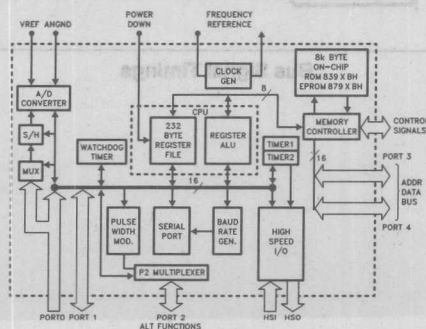
The CPU, as well as the 8- or 16-bit external bus, supports bit, byte, and word operations. Thirty-two-bit double words are supported for a subset of the instruction set. With a 12 MHz input frequency the 8096BH can do a 16-bit addition in 1.0 μ s and a 16 x 16-bit multiply or 32/16 divide in 6.25 μ s. Instruction execution times average 1 μ s to 2 μ s in typical applications.

Four high-speed capture inputs are provided to record the times at which external events occur. Rising, falling, rising and falling, or every eight rising edges can be recorded every 2 μ s at 12 MHz. Six high-speed pulse generator outputs are provided to trigger external events at preset times. The high-speed output unit can simultaneously perform software timer functions. Up to four 16-bit software timers can be in operation at once.

The on-chip A/D converter includes a Sample and Hold, and converts up to 8 multiplexed analog input channels to 10-bit digital values. With a 12 MHz input frequency, each conversion takes 22 μ s. A/D conversions can be performed at preprogrammed times, or asynchronously. This feature is only available on the 8X97BH's.

Also provided on-chip are: a full duplex, double buffered receive serial port with 3 asynchronous and 1 synchronous modes; a 16-bit watchdog timer, and a 256 state pulse-width modulated output signal.

For further information regarding the 8X9XBH product family, consult the MCS-96 section of the Embedded Controller Handbook, or Document number 270090.



270361-1

Figure 1. MCS®-96 Block Diagram

PRODUCT/PACKAGING OPTIONS

The 8X9XBH is available in 68-pin packages, with and without A/D, and with and without on-chip ROM or EPROM. The MCS-96 nomenclature is shown in Figure 2. Figures 4–8 show the pinouts for all available 68-pin packages: Plastic Leaded Chip Carrier (PLCC), Plastic Flatpack (PFP), Ceramic Pin Grid Array (PGA), type "B" Leadless Chip Carrier (LCC), and Ceramic Flatpack (CFP).

The 8X9XBH is also available in three temperature range options: Commercial, Extended and Automotive.

Intel's extended and automotive temperature range products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to 70°C ambient.

With the extended temperature range option, operational characteristics are guaranteed over the temperature range of –40°C to +85°C ambient. For the automotive temperature range option, operational characteristics are guaranteed over the temperature range of –40°C to +125°C case.

The automotive, extended, and commercial temperature versions of the MCS-96 product families are available with or without burn-in options as listed in Table 1.

As shown in Figure 2, temperature, burn-in, and package options are identified by a one- or two-letter prefix to the part number.

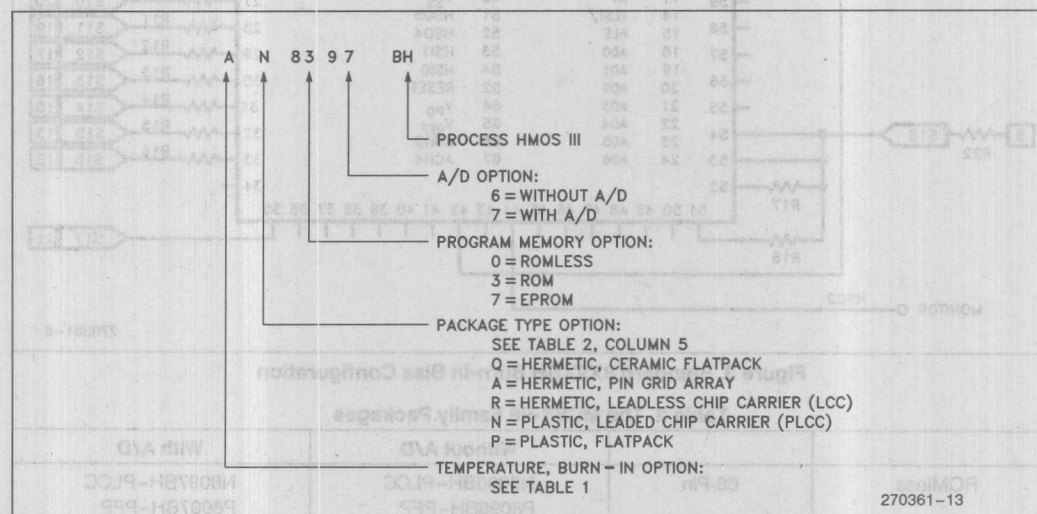


Figure 2. The MCS-96 Family Nomenclature

Table 1. Temperature — Burn-In Option

Temperature Classification	Temperature Designation	Operating Temperature	Burn-In 125°C (Hr)
Commercial	Null Q	0°C to +70°C Ambient 0°C to +70°C Ambient	None 168 ± 8
Extended	T L	–40°C to +85°C Ambient –40°C to +85°C Ambient	None 168 ± 8
Automotive	A B	–40°C to +125°C Case –40°C to +125°C Case	None 168 ± 8

NOTES:

1. Burn-in is dynamic at +125°C, $V_{CC} = 5.5V \pm 0.5V$, following guidelines in MIL-STD-883 method 1015.
2. Other burn-in durations are also available, but not standard.

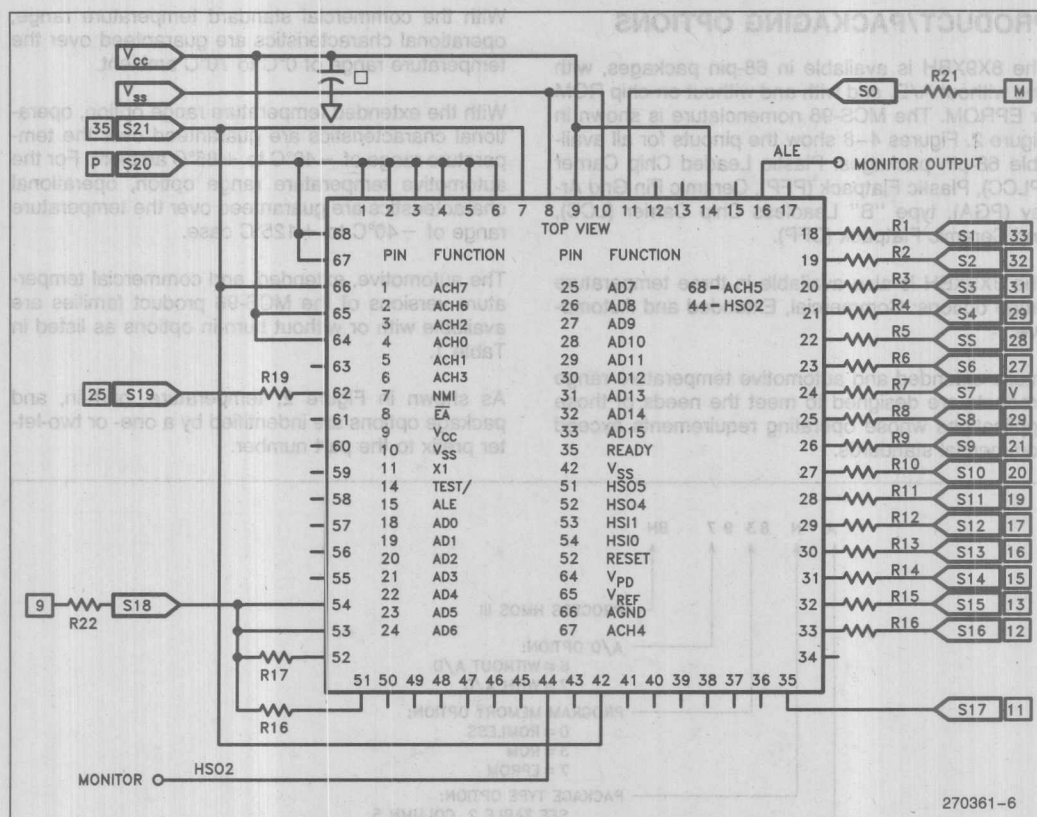


Figure 3. Standard 8X9XBH Burn-In Bias Configuration

Table 2. The MCS[®]-96 Family Packages

		Without A/D	With A/D
ROMless	68-Pin	N8096BH-PLCC P8096BH-PFP	N8097BH-PLCC P8097BH-PFP
ROM (8K Bytes)	68-Pin	N8396BH-PLCC P8396BH-PFP	N8397BH-PLCC P8397BH-PFP
EPROM (8K Bytes)	68-Pin	A8796BH-PGA R8796BH-LCC Q8796BH-CFP	A8797BH-PGA R8797BH-LCC Q8797BH-CFP
None	0°C to +70°C Ambient	None	Commercial
None	-40°C to +85°C Ambient	None	Extended
None	-40°C to +125°C Case	None	Automotive

CFP/ PFP/ PGA/ LCC	PLCC	Description	CFP/ PFP/ PGA/ LCC	PLCC	Description	CFP/ PFP/ PGA/ LCC	PLCC	Description
1	9	ACH7/P0.7/PMOD.3	24	54	AD6/P3.6	47	31	P1.6
2	8	ACH6/P0.6/PMOD.2	25	53	AD7/P3.7	48	30	P1.5
3	7	ACH2/P0.2	26	52	AD8/P4.0	49	29	HSO.1
4	6	ACH0/P0.0	27	51	AD9/P4.1	50	28	HSO.0
5	5	ACH1/P0.1	28	50	AD10/P4.2	51	27	HSO.5/HSI.3
6	4	ACH3/P0.3	29	49	AD11/P4.3	52	26	HSO.4/HSI.2
7	3	NMI	30	48	AD12/P4.4	53	25	HSI.1
8	2	EA	31	47	AD13/P4.5	54	24	HSI.0
9	1	VCC	32	46	AD14/P4.6	55	23	P1.4
10	68	VSS	33	45	AD15/P4.7	56	22	P1.3
11	67	XTAL1	34	44	T2CLK/P2.3	57	21	P1.2
12	66	XTAL2	35	43	READY	58	20	P1.1
13	65	CLKOUT	36	42	T2RST/P2.4	59	19	P1.0
14	64	BUSWIDTH	37	41	BHE/WRH	60	18	TXD/P2.0/PVER/SALE
15	63	INST	38	40	WR/WRL	61	17	RXD/P2.1/PALE
16	62	ALE/ADV	39	39	PWM/P2.5/PD0/SPROG	62	16	RESET
17	61	RD	40	38	P2.7	63	15	EXTINT/P2.2/PROG
18	60	AD0/P3.0	41	37	VPP	64	14	VPD
19	59	AD1/P3.1	42	36	VSS	65	13	VREF
20	58	AD2/P3.2	43	35	HSO.3	66	12	ANGND
21	57	AD3/P3.3	44	34	HSO.2	67	11	ACH4/P0.4/PMOD.0
22	56	AD4/P3.4	45	33	P2.6	68	10	ACH5/P0.5/PMOD.1
23	55	AD5/P3.5	46	32	P1.7			

Figure 4. PGA, PFP, CFP, PLCC and LCC Function Pinouts

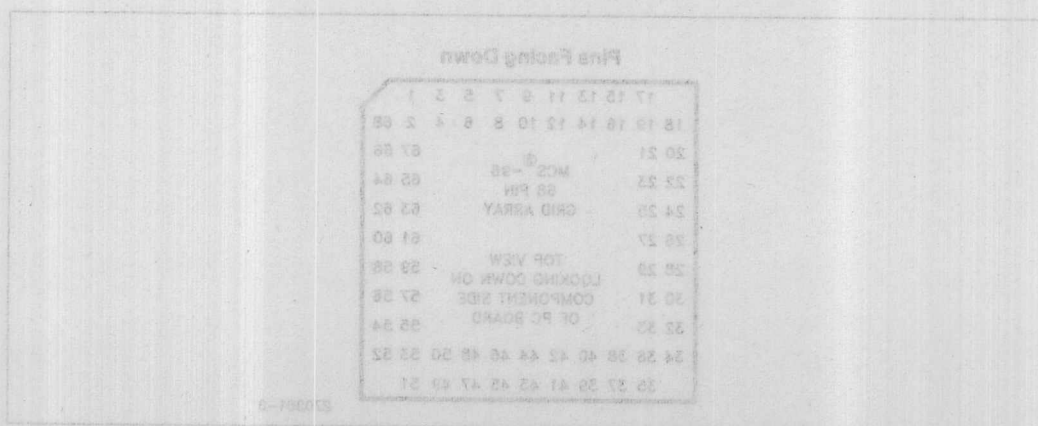
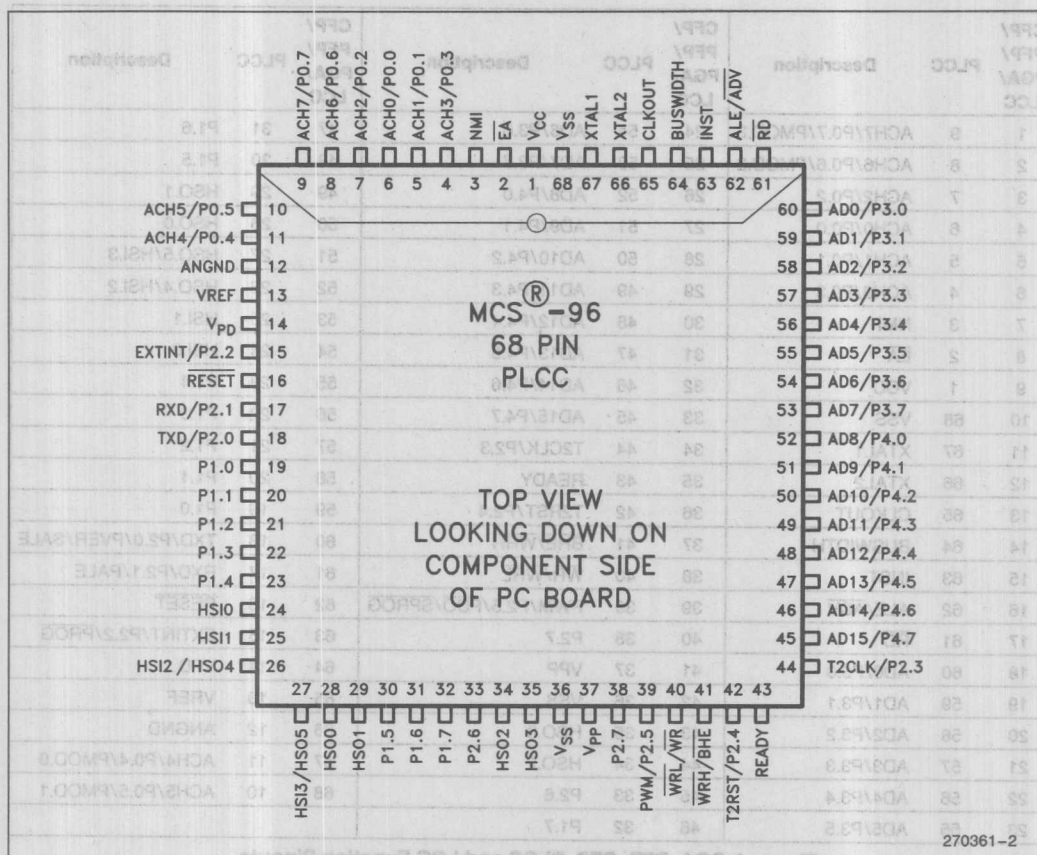


Figure 5. PGA Pinout (Top View)



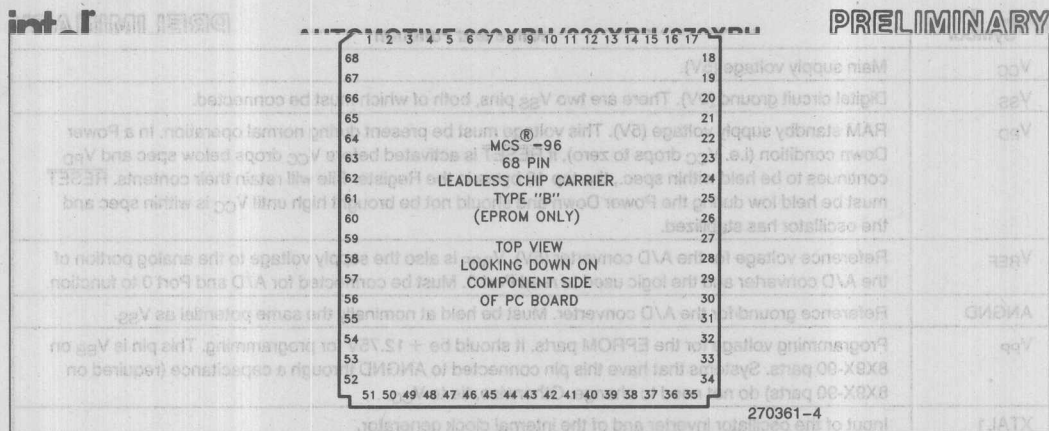


Figure 7. 68-Pin Package (LCC—Top View)

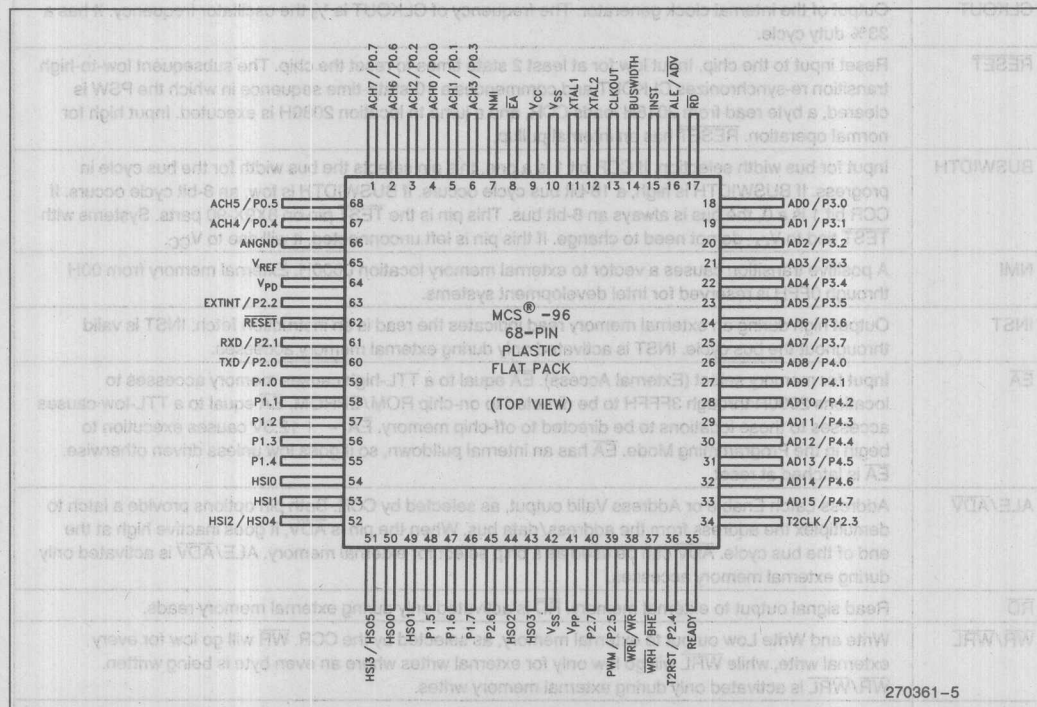


Figure 8. 68-Pin Package (Flat Pack—Top View)

NOTE:

1. When the pin grid array package is mounted on the PC board, the pins are numbered counterclockwise as seen from the component side of the board, just like the flat pack when it's mounted in the contactor. Consequently, the PC board layout for the pin grid array is compatible with the flat pack in a contactor except for the footprint size. The pin functions (from 1 to 68) on both packages are identical. Refer to Intel's Packaging Outlines and Dimensions, Order Number 231369, for mechanical dimensions on these packages.

PIN DESCRIPTIONS

Symbol	Name and Function
V _{CC}	Main supply voltage (5V).
V _{SS}	Digital circuit ground (0V). There are two V _{SS} pins, both of which must be connected.
V _{PD}	RAM standby supply voltage (5V). This voltage must be present during normal operation. In a Power Down condition (i.e. V _{CC} drops to zero), if RESET is activated before V _{CC} drops below spec and V _{PD} continues to be held within spec., the top 16 bytes in the Register File will retain their contents. RESET must be held low during the Power Down and should not be brought high until V _{CC} is within spec and the oscillator has stabilized.
V _{REF}	Reference voltage for the A/D converter (5V). V _{REF} is also the supply voltage to the analog portion of the A/D converter and the logic used to read Port 0. Must be connected for A/D and Port 0 to function.
ANGND	Reference ground for the A/D converter. Must be held at nominally the same potential as V _{SS} .
V _{PP}	Programming voltage for the EPROM parts. It should be +12.75V for programming. This pin is V _{BB} on 8X9X-90 parts. Systems that have this pin connected to ANGND through a capacitance (required on 8X9X-90 parts) do not need to change. Otherwise, tie to V _{CC} .
XTAL1	Input of the oscillator inverter and of the internal clock generator.
XTAL2	Output of the oscillator inverter.
CLKOUT	Output of the internal clock generator. The frequency of CLKOUT is 1/3 the oscillator frequency. It has a 33% duty cycle.
RESET	Reset input to the chip. Input low for at least 2 state times to reset the chip. The subsequent low-to-high transition re-synchronizes CLKOUT and commences a 10-state-time sequence in which the PSW is cleared, a byte read from 2018H loads CCR, and a jump to location 2080H is executed. Input high for normal operation. RESET has an internal pullup.
BUSWIDTH	Input for bus width selection. If CCR bit 1 is a one, this pin selects the bus width for the bus cycle in progress. If BUSWIDTH is high, a 16-bit bus cycle occurs. If BUSWIDTH is low, an 8-bit cycle occurs. If CCR bit 1 is a 0, the bus is always an 8-bit bus. This pin is the TEST pin on 8X9X-90 parts. Systems with TEST tied to V _{CC} do not need to change. If this pin is left unconnected, it will rise to V _{CC} .
NMI	A positive transition causes a vector to external memory location 0000H. External memory from 00H through 0FFH is reserved for Intel development systems.
INST	Output high during an external memory read indicates the read is an instruction fetch. INST is valid throughout the bus cycle. INST is activated only during external memory accesses.
EA	Input for memory select (External Access). EA equal to a TTL-high causes memory accesses to locations 2000H through 3FFFH to be directed to on-chip ROM/EPROM. EA equal to a TTL-low causes accesses to these locations to be directed to off-chip memory. EA = +12.5V causes execution to begin in the Programming Mode. EA has an internal pulldown, so it goes low unless driven otherwise. EA is latched at reset.
ALE/ADV	Address Latch Enable or Address Valid output, as selected by CCR. Both pin options provide a latch to demultiplex the address from the address/data bus. When the pin is ADV, it goes inactive high at the end of the bus cycle. ADV can be used as a chip select for external memory. ALE/ADV is activated only during external memory accesses.
RD	Read signal output to external memory. RD is activated only during external memory reads.
WR/WRL	Write and Write Low output to external memory, as selected by the CCR. WR will go low for every external write, while WRL will go low only for external writes where an even byte is being written. WR/WRL is activated only during external memory writes.
BHE/WRH	Bus High Enable or Write High output to external memory, as selected by the CCR. BHE low selects the bank of memory that is connected to the high byte of the data bus. A0 low selects the bank of memory that is connected to the low byte of the data bus. Thus accesses to a 16-bit wide memory can be to the low byte only (A0 low, BHE high), to the high byte only (A0 high, BHE low), or both bytes (A0 low, BHE low). If the WRH function is selected, the pin will go low if the bus cycle is writing to an odd memory location. BHE/WRH is activated only during external memory writes.

PIN DESCRIPTIONS (Continued)

Symbol	Name and Function
READY	Ready input to lengthen external memory cycles, for interfacing to slow or dynamic memory, or for bus sharing. If the pin is high, CPU operation continues in a normal manner. If the pin is low prior to the falling edge of CLKOUT, the memory controller goes into a wait mode until the next positive transition in CLKOUT occurs with READY high. The bus cycle can be lengthened by up to 1 μ s. When the external memory is not being used, READY has no effect. Internal control of the number of wait states inserted into a bus cycle held not ready is available through configuration of CCR. READY has a weak internal pullup, so it goes high unless externally pulled low.
HSI	Inputs to High Speed Input Unit. Four HSI pins are available: HSI.0, HSI.1, HSI.2, and HSI.3. Two of them (HSI.2 and HSI.3) are shared with the HSO Unit. The HSI pins are also used as inputs by EPROM parts in Programming Mode.
HSO	Outputs from High Speed Output Unit. Six HSO pins are available: HSO.0, HSO.1, HSO.2, HSO.3, HSO.4, and HSO.5. Two of them (HSO.4 and HSO.5) are shared with the HSI Unit.
Port 0	8-bit high impedance input-only port. These pins can be used as digital inputs and/or as analog inputs to the on-chip A/D converter. These pins are also a mode input to EPROM parts in the Programming Mode.
Port 1	8-bit quasi-bidirectional I/O port.
Port 2	8-bit multi-functional port. Six of its pins are shared with other functions in the 8096BH, the remaining 2 are quasi-bidirectional. These pins are also used to input and output control signals on EPROM parts in Programming Mode.
Ports 3 and 4	8-bit bi-directional I/O ports with open drain outputs. These pins are shared with the multiplexed address/data bus which has strong internal pullups. Ports 3 and 4 are also used as a command, address and data path by EPROM parts operating in the Programming Mode.

Symbol	Parameter	Min	Max	Units	Test Conditions
I_{CC1}	V _{CC} Supply Current -40°C \leq +125°C		270	mA	All Outputs Disconnected
I_{CC2}	V _{CC} Supply Current (T _A = +70°C)		185	mA	
I_{CC3}	V _{CC} Supply Current		1	mA	Normal Operation and Power Down
I_{CC4}	V _{CC} Supply Current		10	mA	
V_{IL}	Input Low Voltage (Except RESET)	-0.8	+0.8	V	
V_{IH}	Input High Voltage, RESET	-0.3	+0.7	V	
V_{IH}	Input High Voltage (Except RESET, NM, XTAL1)	2.0	V _{CC} + 0.5	V	
V_{IH}	Input High Voltage, RESET Rising	2.4	V _{CC} + 0.5	V	
V_{IH}	Input High Voltage, RESET Falling	2.1	V _{CC} + 0.5	V	
V_{IH}	Input High Voltage, NM, XTAL1	2.3	V _{CC} + 0.5	V	
I_{L1}	Input Leakage Current to each pin of HSI, P3, P4, and P5.1	± 10		nA	V _{IN} = 0 to V _{CC}
I_{L2}	D.C. Input Leakage Current to each pin of P0	± 3		nA	V _{IN} = 0 to V _{CC}
I_{L3}	Input High Current to EA	100		nA	V _{IN} = 2.5V
I_{L4}	Input Low Current to each pin of P1, and to P2.6, P2.7	-150		nA	V _{IN} = 0.45V
I_{L5}	Input Low Current to RESET	-5		mA	V _{IN} = 0.45V
I_{L6}	Input Low Current P2.2, P2.3, P2.4, READY, BUSWIDTH	-80		nA	V _{IN} = 0.45V
V_{OL}	Output Low Voltage on Quasi-Bidirectional	0.45		V	I _{OL} = 0.8 mA (Note 1)
V_{OH1}	Output Low Voltage on Quasi-Bidirectional	0.75		V	I _{OL} = 2.0 mA (Notes 1, 2, 3)
V_{OH2}	Output Low Voltage on Standard Output	0.45		V	I _{OL} = 0.8 mA (Note 1)

ABSOLUTE MAXIMUM RATINGS*

Case Temperature Under Bias	−40°C to +125°C
Storage Temperature	−40°C to +150°C
Voltage from V _{PP} to V _{SS} or ANGND	+13.0V
Voltage from Any Other Pin to V _{SS} or ANGND	−0.3V to +7.0V
Average Output Current from Any Pin	10 mA
Power Dissipation	1.5W

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
T _C	Case Temperature Under Bias	−40	+125	°C
V _{CC}	Digital Supply Voltage	4.50	5.50	V
V _{REF}	Analog Supply Voltage	4.50	5.50	V
f _{OSC}	Oscillator Frequency	6.0	12.0	MHz
V _{PD}	Power Down Supply Voltage	4.50	5.50	V

NOTE:

ANGND and V_{SS} should be nominally at the same potential.

D.C. CHARACTERISTICS Test Conditions: V_{CC}, V_{REF}, V_{PD}, V_{PP}, V_{SS} = 5.0V ± 0.5V; f_{OSC} = 6.0 MHz to 12.0 MHz; T_C = −40°C to +125°C; V_{SS}, ANGND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
I _{CC}	V _{CC} Supply Current −40°C ≤ +125°C		270	mA	All Outputs Disconnected
I _{CC1}	V _{CC} Supply Current (T _A = +70°C)		185	mA	
I _{PD}	V _{PD} Supply Current		1	mA	Normal Operation and Power Down
I _{REF}	V _{REF} Supply Current		10	mA	
V _{IL}	Input Low Voltage (Except RESET)	−0.3	+0.8	V	
V _{IL1}	Input Low Voltage, RESET	−0.3	+0.7	V	
V _{IH}	Input High Voltage (Except RESET, NMI, XTAL1)	2.0	V _{CC} + 0.5	V	
V _{IH1}	Input High Voltage, RESET Rising	2.4	V _{CC} + 0.5	V	
V _{IH2}	Input High Voltage, RESET Falling	2.1	V _{CC} + 0.5	V	
V _{IH3}	Input High Voltage, NMI, XTAL1	2.3	V _{CC} + 0.5	V	
I _{LI}	Input Leakage Current to each pin of HSI, P3, P4, and to P2.1.		±10	μA	V _{IN} = 0 to V _{CC}
I _{LI1}	D.C. Input Leakage Current to each pin of P0		+3	μA	V _{IN} = 0 to V _{CC}
I _{IH}	Input High Current to EA		100	μA	V _{IH} = 2.4V
I _{IL}	Input Low Current to each pin of P1, and to P2.6, P2.7.		−150	μA	V _{IL} = 0.45V
I _{IL1}	Input Low Current to RESET	−0.25	−2	mA	V _{IL} = 0.45V
I _{IL2}	Input Low Current P2.2, P2.3, P2.4, READY, BUSWIDTH		−50	μA	V _{IL} = 0.45V
V _{OL}	Output Low Voltage on Quasi-Bidirectional port pins and P3, P4 when used as ports		0.45	V	I _{OL} = 0.8 mA (Note 1)
V _{OL1} (4)	Output Low Voltage on Quasi-Bidirectional port pins and P3, P4 when used as ports		0.75	V	I _{OL} = 2.0 mA (Notes 1, 2, 3)
V _{OL2}	Output Low Voltage on Standard Output pins, RESET and Bus/Control Pins		0.45	V	I _{OL} = 0.8 mA (Note 1)

D.C. CHARACTERISTICS Test Conditions: $V_{CC}, V_{REF}, V_{PD}, V_{PP}, V_{SS} = 5.0V \pm 0.5V$;
 $f_{OSC} = 6.0 \text{ MHz to } 12.0 \text{ MHz}$; $T_C = -40^\circ\text{C to } +125^\circ\text{C}$; $V_{SS}, \text{ANGND} = 0V$ (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{OH}	Output High Voltage on Quasi-Bidirectional pins	2.4		V	$I_{OH} = -20 \mu A$ (Note 1)
V_{OH1}	Output High Voltage on Standard Output pins and Bus/Control pins	2.4		V	$I_{OH} = -200 \mu A$ (Note 1)
I_{OH3}	Output High Current on RESET	-50		μA	$V_{OH} = 2.4V$
C_S	Pin Capacitance (Any Pin to V_{SS})		10	pF	$f_{TEST} = 1.0 \text{ MHz}$

NOTES:

- Quasi-bidirectional pins include those on P1, for P2.6 and P2.7. Standard Output Pins include TXD, RXD (Mode 0 only), PWM, and HSO pins. Bus/Control pins include CLKOUT, ALE, BHE, RD, WR, INST and AD0-15.
- Maximum current per pin must be externally limited to the following values if V_{OL} is held above 0.45V.
 I_{OL} on quasi-bidirectional pins and Ports 3 and 4 when used as ports: 4.0 mA
 I_{OL} on standard output pins and RESET: 8.0 mA
 I_{OL} on Bus/Control pins: 2.0 mA
 I_{OL} on HSO.x = 1.6 mA: x = 0, 4, and 5 @ $V_{OL} = 0.5V$
- During normal (non-transient) operation the following limits apply:
 Total I_{OL} on Port 1 must not exceed 8.0 mA.
 Total I_{OL} on P2.0, P2.6, RESET and all HSO pins must not exceed 15 mA.
 Total I_{OL} on Port 3 must not exceed 10 mA.
 Total I_{OL} on P2.5, P2.7, and Por: 4 must not exceed 20 mA.
- Only tested on 879XBH devices.

A.C. CHARACTERISTICS

$V_{CC}, V_{PD} = 4.5V \text{ to } 5.5V$; $T_C = -40^\circ\text{C to } +125^\circ\text{C}$; $f_{OSC} = 6.0 \text{ MHz to } 12.0 \text{ MHz}$

Test Conditions: Load Capacitance on Output Pins = 80 pF
 Oscillator Frequency = 12 MHz

TIMING REQUIREMENTS (Other system components must meet these specifications)

Symbol	Parameter	Min	Max	Units
T_{CLYX}	READY Hold after CLKOUT Edge	0		ns
T_{LLYV}	End of ALE/ADV to READY Valid		$2T_{osc} - 70$	ns
T_{LLYH}	End of ALE/ADV to READY High	$2T_{osc} + 40$	$4T_{osc} - 80$	ns
$T_{YLYH}^{(2)}$	Non-Ready Time		1000	ns
$T_{AVDV}^{(1)}$	Address Valid to Input Data Valid		$5T_{osc} - 120$	ns
T_{RLDV}	RD Active to Input Data Valid		$3T_{osc} - 100$	ns
T_{RHDV}	Data Hold after RD Inactive	0		ns
$T_{RHDZ}^{(2)}$	RD Inactive to Input Data Float	0	$T_{osc} - 25$	ns
$T_{AVGV}^{(1)}$	Address Valid to BUSWIDTH Valid		$2T_{osc} - 125$	ns
T_{LLGX}	BUSWIDTH Hold after ALE/ADV Low	$T_{osc} + 40$		ns
T_{LLGV}	ALE/ADV Low to BUSWIDTH Valid		$T_{osc} - 75$	ns

NOTES:

- The term "Address Valid" applies to AD0-AD15, BHE and INST.
- Not currently tested in production.

A.C. CHARACTERISTICS (Continued)

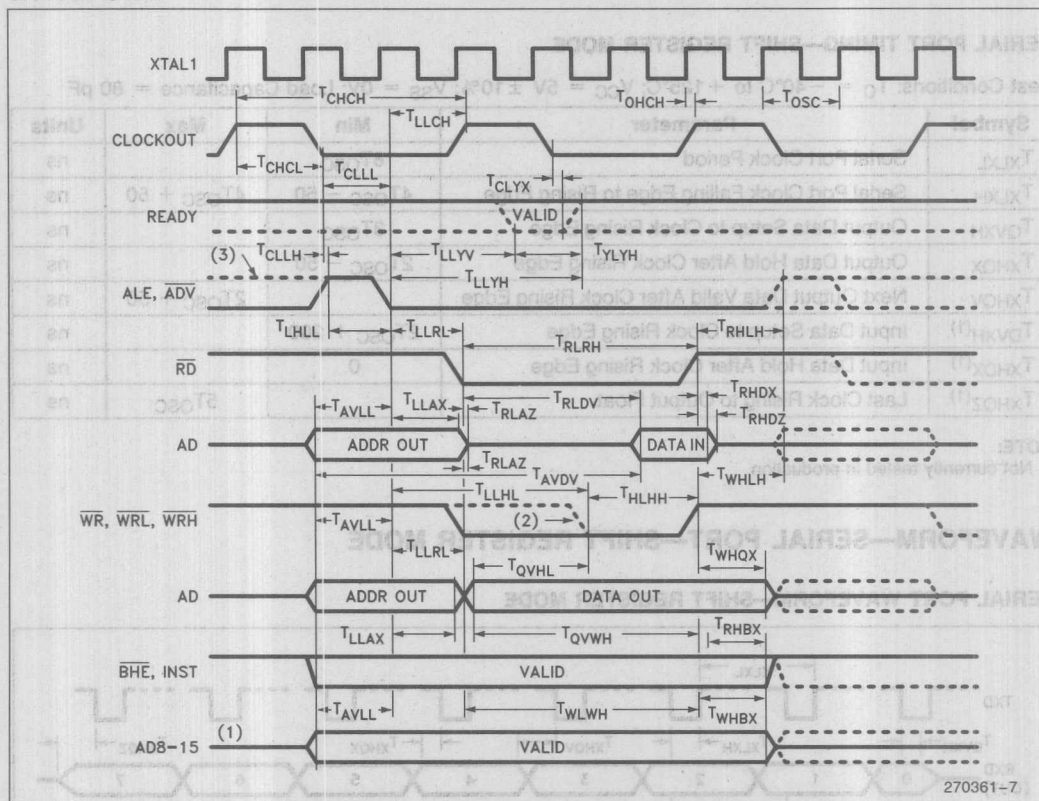
TIMING RESPONSES (MCS-96 parts meet these specifications)

Symbol	Parameter	Min	Max	Units
F _{XTAL}	Oscillator Frequency	6.0	12.0	MHz
T _{Osc}	Oscillator Period	83	166	ns
T _{OHCH}	XTAL1 Rising Edge to Clockout Rising Edge	0(4)	120(4)	ns
T _{CHCH}	CLKOUT Period(4)	3Tosc(4)	3Tosc(4)	ns
T _{CHCL}	CLKOUT High Time	Tosc - 35	Tosc + 10	ns
T _{CLLH}	CLKOUT Low to ALE High	-20	35	ns
T _{LLCH}	ALE/ADV Low to CLKOUT High	Tosc - 25	Tosc + 45	ns
T _{LHLL}	ALE/ADV High Time	Tosc - 30	Tosc + 15(5)	ns
T _{AVLL} (1)	Address Setup to End of ALE/ADV	Tosc - 50		ns
T _{RLAZ} (6)	RD or WR Low to Address Float		25	ns
T _{LLRL}	End of ALE/ADV to RD or WR Active	Tosc - 40		ns
T _{LLAX} (6)	Address Hold after End of ALE/ADV	Tosc - 40		ns
T _{WLWH}	WR Pulse Width	3Tosc - 35		ns
T _{QVWH}	Output Data Valid to End of WR/WRL/WRH	3Tosc - 60		ns
T _{WHQX}	Output Data Hold after WR/WRL/WRH	Tosc - 50		ns
T _{WHLH}	End of WR/WRL/WRH to ALE/ADV High	Tosc - 75		ns
T _{RLRH}	RD Pulse Width	3Tosc - 30		ns
T _{RHLH}	End of RD to ALE/ADV High	Tosc - 45		ns
T _{CLLL}	CLOCKOUT Low to ALE/ADV Low	Tosc - 40	Tosc + 35	ns
T _{RHBX}	RD High to INST, BHE, AD8-15 Inactive	Tosc - 25	Tosc + 30	ns
T _{WHBX}	WR High to INST, BHE, AD8-15 Inactive	Tosc - 50	Tosc + 100	ns
T _{HLHH}	WRL, WRH Low to WRL, WRH High	2Tosc - 35	2Tosc + 40	ns
T _{LLHL}	ALE/ADV Low to WRL, WRH Low	2Tosc - 30	2Tosc + 55	ns
T _{QVHL}	Output Data Valid to WRL, WRH Low	Tosc - 60		ns

NOTES:

1. The term "Address Valid" applies to AD0-15, BHE and INST.
2. Not Currently tested in production.
3. If more than one wait state is desired, add 3Tosc for each additional wait state.
4. CLKOUT is directly generated as a divide by 3 of the oscillator. The period will be 3Tosc ± 10 ns if TOSC is constant and the rise and fall times on XTAL1 are less than 10 ns.
5. Max spec applies only to ALE. Min spec applies to both ALE and ADV.
6. The term "Address" in this definition applies to AD0-7 for 8-bit cycles, and AD0-15 for 16-bit cycles.

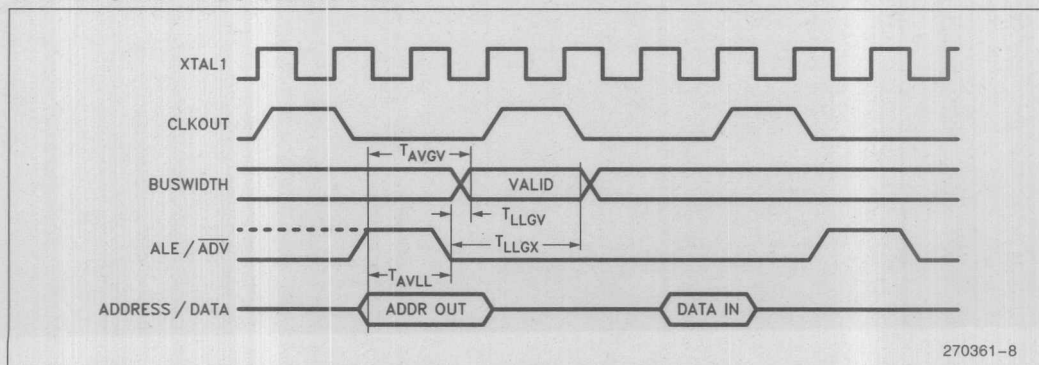
WAVEFORM



NOTES:

1. 8-bit bus only.
2. When write strobe mode selected.
3. When ADV selected.

WAVEFORM—BUSWIDTH PIN



A.C. CHARACTERISTICS—SERIAL PORT—SHIFT REGISTER MODE

SERIAL PORT TIMING—SHIFT REGISTER MODE

Test Conditions: $T_C = -40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$; Load Capacitance = 80 pF

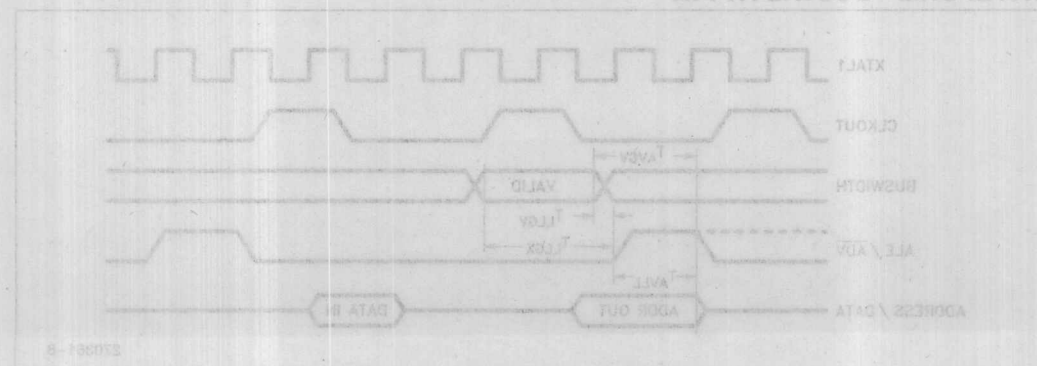
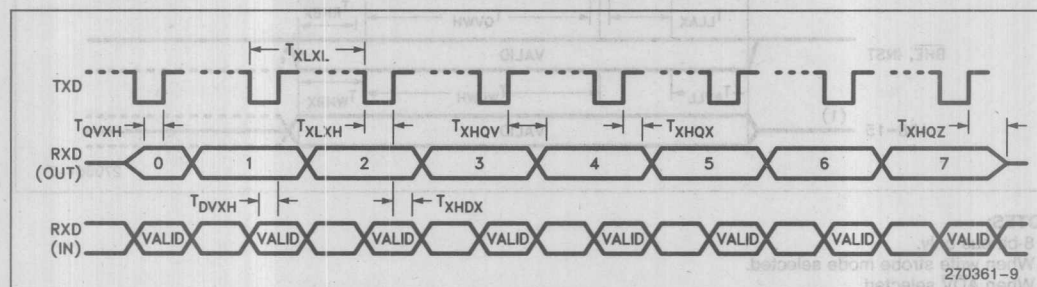
Symbol	Parameter	Min	Max	Units
T_{XLXL}	Serial Port Clock Period	$8T_{OSC}$		ns
T_{XLXH}	Serial Port Clock Falling Edge to Rising Edge	$4T_{OSC} - 50$	$4T_{OSC} + 50$	ns
T_{QVXH}	Output Data Setup to Clock Rising Edge	$3T_{OSC}$		ns
T_{XHGX}	Output Data Hold After Clock Rising Edge	$2T_{OSC} - 50$		ns
T_{XHGV}	Next Output Data Valid After Clock Rising Edge		$2T_{OSC} + 50$	ns
$T_{DVXH}^{(1)}$	Input Data Setup to Clock Rising Edge	$2T_{OSC} + 200$		ns
$T_{XHDX}^{(1)}$	Input Data Hold After Clock Rising Edge	0		ns
$T_{XHGX}^{(1)}$	Last Clock Rising to Output Float		$5T_{OSC}$	ns

NOTE:

1. Not currently tested in production.

WAVEFORM—SERIAL PORT—SHIFT REGISTER MODE

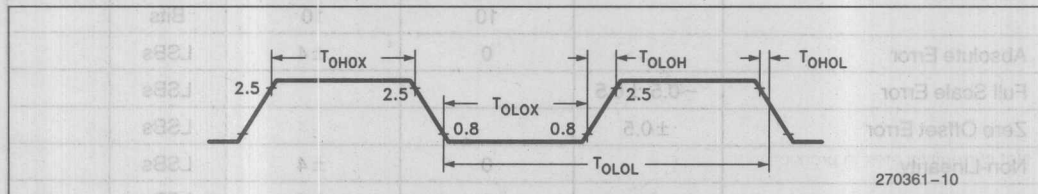
SERIAL PORT WAVEFORM—SHIFT REGISTER MODE



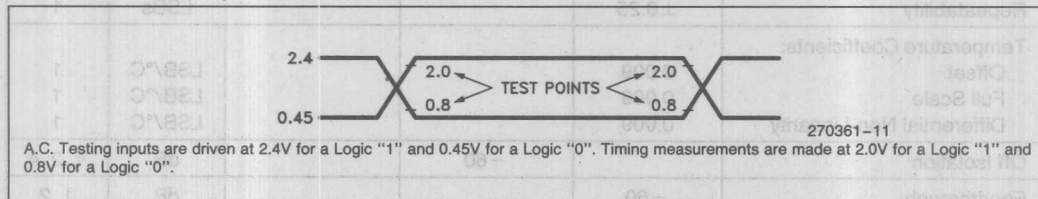
EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/T _{OLOL}	Oscillator Frequency	6	12	MHz
T _{OH0X}	High Time	25		ns
T _{OLOX}	Low Time	25		ns
T _{OLOH}	Rise Time		15	ns
T _{OHOL}	Fall Time		15	ns

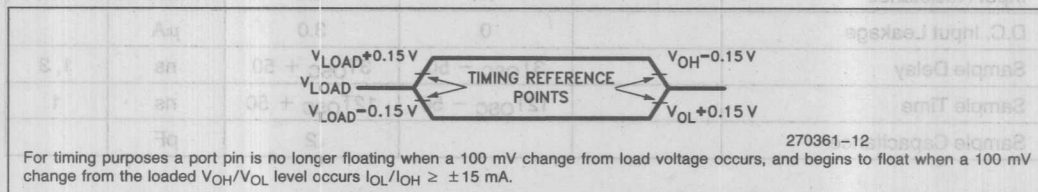
EXTERNAL CLOCK DRIVE WAVEFORMS



A.C. TESTING INPUT, OUTPUT WAVEFORM



FLOAT WAVEFORM



A/D CONVERTER SPECIFICATIONS

OPERATING CONDITIONS

A/D Converter operation is verified only on the 8097BH, 8397BH, 8797BH.

The absolute conversion accuracy is dependent on the accuracy of V_{REF} . The specifications given below assume adherence to the Operating Conditions section of these data sheets. Testing is done at $V_{REF} = 5.120V$.

V_{CC}, V_{PD}, V_{REF}	4.5V to 5.5V
$V_{SS}, ANGND$	0.0V
T_C	-40°C to +125°C
F_{OSC}	6.0 to 12.0 MHz
Test Conditions:	
V_{REF}	5.120V

Parameter	Typical*(1)	Minimum	Maximum	Units**	Notes
Resolution		1024	1024	Levels	
		10	10	Bits	
Absolute Error		0	±4	LSBs	
Full Scale Error	-0.5 ±0.5			LSBs	
Zero Offset Error	±0.5			LSBs	
Non-Linearity		0	±4	LSBs	
Differential Non-Linearity		0	±2	LSBs	
Channel-to-Channel Matching		0	±1	LSBs	
Repeatability	±0.25			LSBs	1
Temperature Coefficients:					
Offset	0.009			LSB/°C	1
Full Scale	0.009			LSB/°C	1
Differential Non-Linearity	0.009			LSB/°C	1
Off Isolation		-60		dB	1, 2, 4
Feedthrough	-60			dB	1, 2
V_{CC} Power Supply Rejection	-60			dB	1, 2
Input Resistance		1K	5K	Ω	1
D.C. Input Leakage		0	3.0	μA	
Sample Delay		$3T_{OSC} - 50$	$3T_{OSC} + 50$	ns	1, 3
Sample Time		$12T_{OSC} - 50$	$12T_{OSC} + 50$	ns	1
Sample Capacitance			2	pF	

NOTES:

* These values are expected for most parts at 25°C.

** An "LSB", as used here, is defined in the glossary which follows and has a value of approximately 5 mV.

1. These values are not tested in production and are based on theoretical estimates and laboratory tests.

2. DC to 100 KHz.

3. For starting the A/D with an HSO Command.

4. Multiplexer Break-Before-Make Guaranteed.

EPROM SPECIFICATIONS

A.C. EPROM PROGRAMMING CHARACTERISTICS

Operating Conditions: Load Capacitance = 150 pF, $T_A = 25^\circ\text{C}, \pm 5^\circ\text{C}$, $V_{CC}, V_{PD}, V_{REF} = 5.0\text{V} \pm 0.5\text{V}$, $V_{SS}, \text{AGND} = 0\text{V}$, $V_{PP} = 12.75\text{V} \pm 0.25\text{V}$, $\overline{\text{EA}} = 11\text{V} \pm 2.0\text{V}$, $f_{\text{OSC}} = 6.0\text{ MHz}$

Symbol	Parameter	Min	Max	Units
T_{AVLL}	ADDRESS/COMMAND Valid to PALE Low	0		T_{OSC}
T_{LLAX}	ADDRESS/COMMAND Hold After PALE Low	80		T_{OSC}
T_{DVPL}	Output Data Setup Before $\overline{\text{PROG}}$ Low	0		T_{OSC}
T_{PLDX}	Data Hold After $\overline{\text{PROG}}$ Falling	80		T_{OSC}
T_{LLH}	PALE Pulse Width	180		T_{OSC}
T_{PLPH}	$\overline{\text{PROG}}$ Pulse Width	$250 T_{\text{OSC}}$	$100 \mu\text{s} + 144 T_{\text{OSC}}$	
T_{LHPL}	PALE High to $\overline{\text{PROG}}$ Low	250		T_{OSC}
T_{PHLL}	$\overline{\text{PROG}}$ High to Next PALE Low	600		T_{OSC}
T_{PHDX}	Data Hold After $\overline{\text{PROG}}$ High	30		T_{OSC}
T_{PHVV}	$\overline{\text{PROG}}$ High to $\overline{\text{PVER}}/\overline{\text{PDO}}$ Valid	500		T_{OSC}
T_{LLVH}	PALE Low to $\overline{\text{PVER}}/\overline{\text{PDO}}$ High	100		T_{OSC}
T_{PLDV}	$\overline{\text{PROG}}$ Low to VERIFICATION/DUMP Data Valid	100		T_{OSC}
T_{SHLL}	RESET High to First PALE Low (not shown)	2000		T_{OSC}

NOTE:

Run-time programming is done with $f_{\text{OSC}} = 6.0\text{ MHz}$ to 12.0 MHz , $V_{CC}, V_{PD}, V_{REF} = 5\text{V} \pm 0.5\text{V}$, $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$ and $V_{PP} = 12.75\text{V} \pm 0.25\text{V}$. For run-time programming over a full operating range, contact the factory.

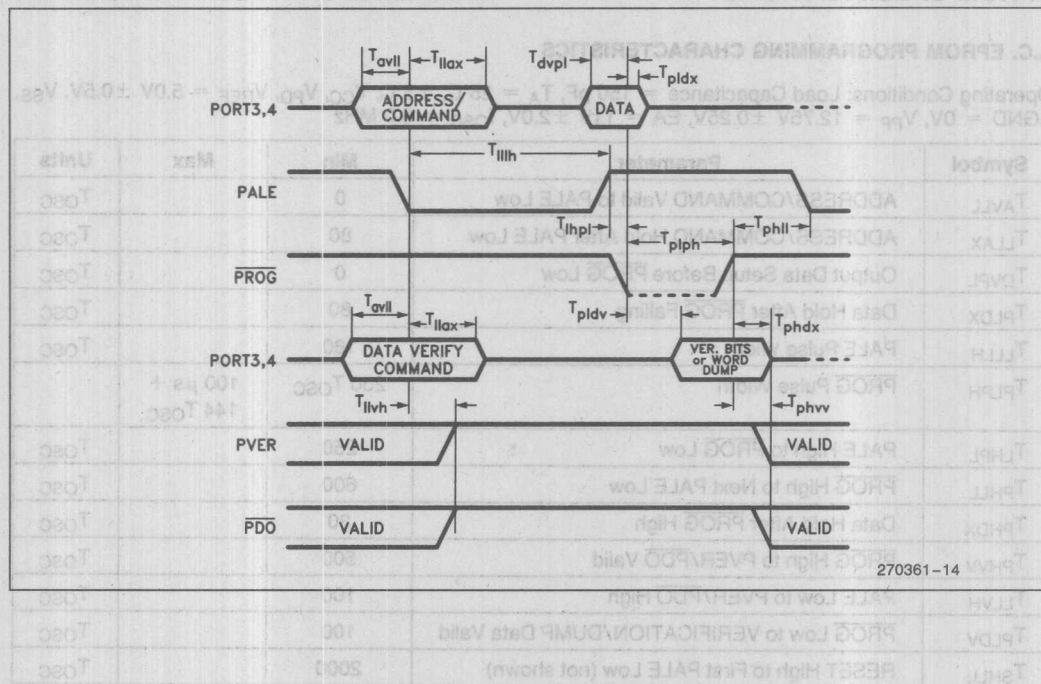
D.C. EPROM PROGRAMMING CHARACTERISTICS

Symbol	Parameter	Min	Max	Units
I_{PP}	V_{PP} Supply Current (Whenever Programming)		100	mA

NOTE:

V_{PP} must be within 1V of V_{CC} while $V_{\text{CC}} < 4.5\text{V}$. V_{PP} must not have a low impedance path to ground or V_{SS} while $V_{\text{CC}} > 4.5\text{V}$.

WAVEFORM-EPROM PROGRAMMING



270361-14

NOTE: Run-time programming is done with $F_{osc} = 8.0 \text{ MHz}$ to 12.0 MHz . V_{cc} , V_{pp} , $V_{ref} = 5V \pm 0.5V$, $T_A = 25^\circ C$ ± 10°C and $V_{pp} = 12.75V \pm 0.55V$. For run-time programming over a full operating range, contact the factory.

D.C. EPROM PROGRAMMING CHARACTERISTICS

Symbol	Parameter	Min	Max	Units
I_{pp}	V_{pp} Supply Current (Whenever Programming)		100	mA

NOTE: V_{pp} must be within 1V of V_{cc} while $V_{cc} < 4.5V$. V_{pp} must not have a low impedance path to ground or V_{cc} while $V_{cc} > 4.5V$.

High speed digital signals are frequently encountered in modern control applications. In addition, there is often a requirement for high speed 16-bit and 32-bit operations in calculations. The MCS-96 product line, generically referred to as the 8096, is designed to be used in applications which require high speed calculations and fast I/O operations.

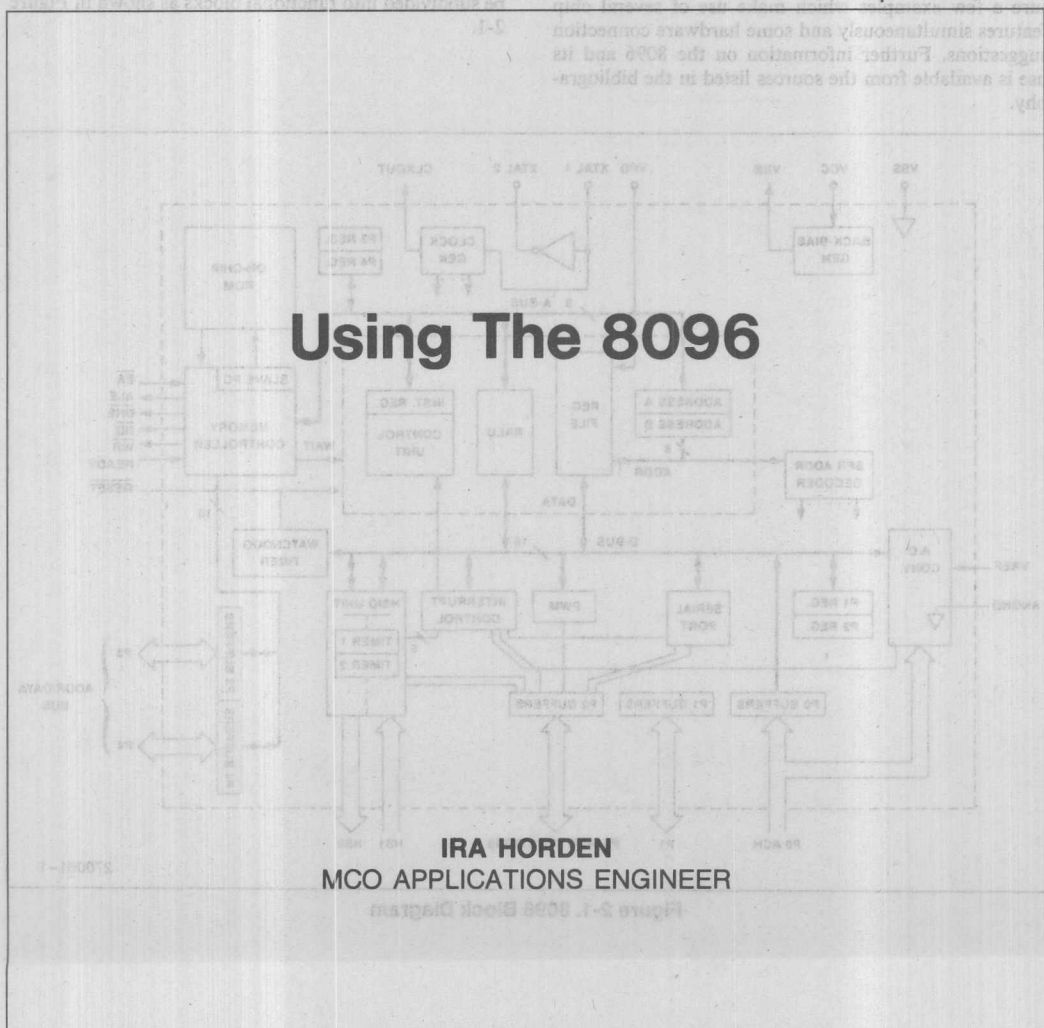
The 8096 is a 16-bit microcontroller with dedicated I/O subsystems and a complete set of 16-bit arithmetic instructions including multiply and divide operations. This Appendix will briefly describe the 8096 in section 1, and then give short examples of how to use each of its features simultaneously and some hardware connection suggestions. Further information on the 8096 and its use is available from the sources listed in the bibliography.

September 1987

2.1 General Description

Unlike microprocessors, microcontrollers are generally optimized for specific applications. Intel's 8096 was optimized for general control tasks while the 8051 was optimized for 8-bit math and single bit boolean operations. The 8096 has been designed for high speed/high performance control applications. Because it has been designed for these applications, the 8096 architecture is different from that of the 8048 or 8051.

There are two major sections of the 8096: the CPU section and the I/O section. Each of these sections are subdivided into functional blocks as shown in Figure 2-1.



Order Number: 270061-002

1.0 INTRODUCTION

High speed digital signals are frequently encountered in modern control applications. In addition, there is often a requirement for high speed 16-bit and 32-bit precision in calculations. The MCS[®]-96 product line, generically referred to as the 8096, is designed to be used in applications which require high speed calculations and fast I/O operations.

The 8096 is a 16-bit microcontroller with dedicated I/O subsystems and a complete set of 16-bit arithmetic instructions including multiply and divide operations. This Ap-note will briefly describe the 8096 in section 2, and then give short examples of how to use each of its key features in section 3. The concluding sections feature a few examples which make use of several chip features simultaneously and some hardware connection suggestions. Further information on the 8096 and its use is available from the sources listed in the bibliography.

2.0 8096 OVERVIEW

2.1. General Description

Unlike microprocessors, microcontrollers are generally optimized for specific applications. Intel's 8048 was optimized for general control tasks while the 8051 was optimized for 8-bit math and single bit boolean operations. The 8096 has been designed for high speed/high performance control applications. Because it has been designed for these applications the 8096 architecture is different from that of the 8048 or 8051.

There are two major sections of the 8096; the CPU section and the I/O section. Each of these sections can be subdivided into functional blocks as shown in Figure 2-1.

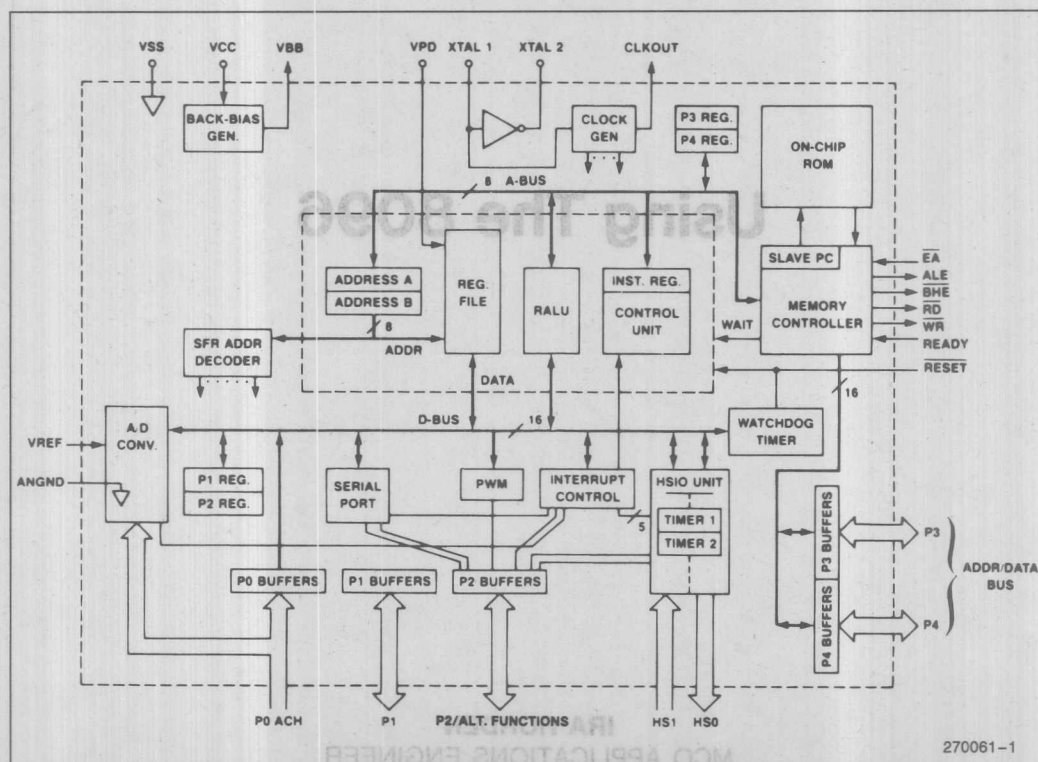


Figure 2-1. 8096 Block Diagram

2.1.1. CPU SECTION

The CPU of the 8096 uses a 16-bit ALU which operates on a 256-byte register file instead of an accumulator. Any of the locations in the register file can be used for sources or destinations for most of the instructions. This is called a register to register architecture. Many of the instructions can also use bytes or words from anywhere in the 64K byte address space as operands. A memory map is shown in Figure 2-2.

In the lower 24 bytes of the register file are the register-mapped I/O control locations, also called Special Function Registers or SFRs. These registers are used to control the on-chip I/O features. The remaining 232 bytes are general purpose RAM, the upper 16 of which can be kept alive using a low current power-down mode.

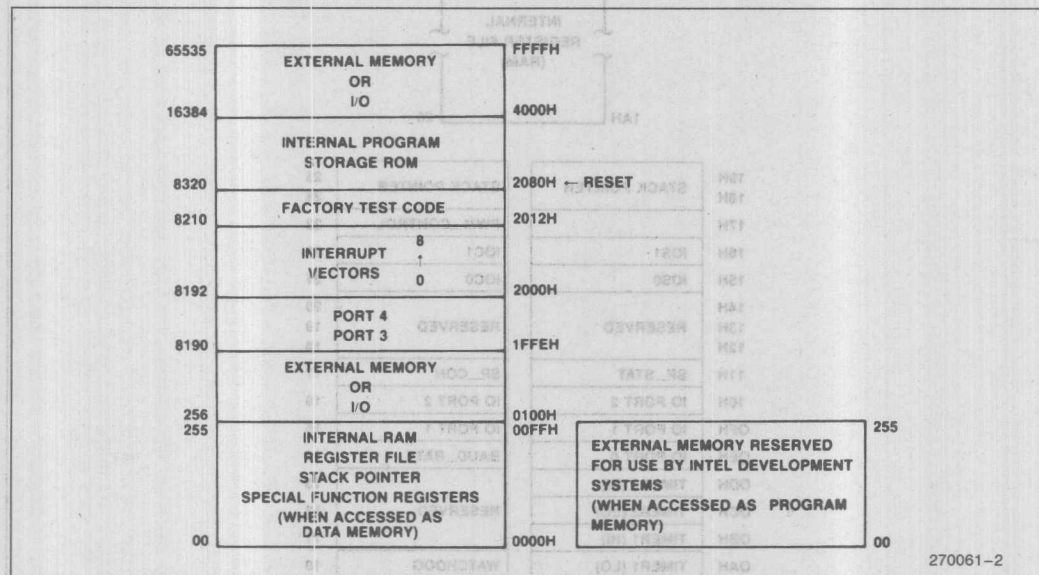


Figure 2-2. Memory Map

Figure 2-3 shows the layout of the register mapped I/O. Some of these registers serve two functions, one if they are read from and another if they are written

to. More information about the use of these registers is included in the description of the features which they control.

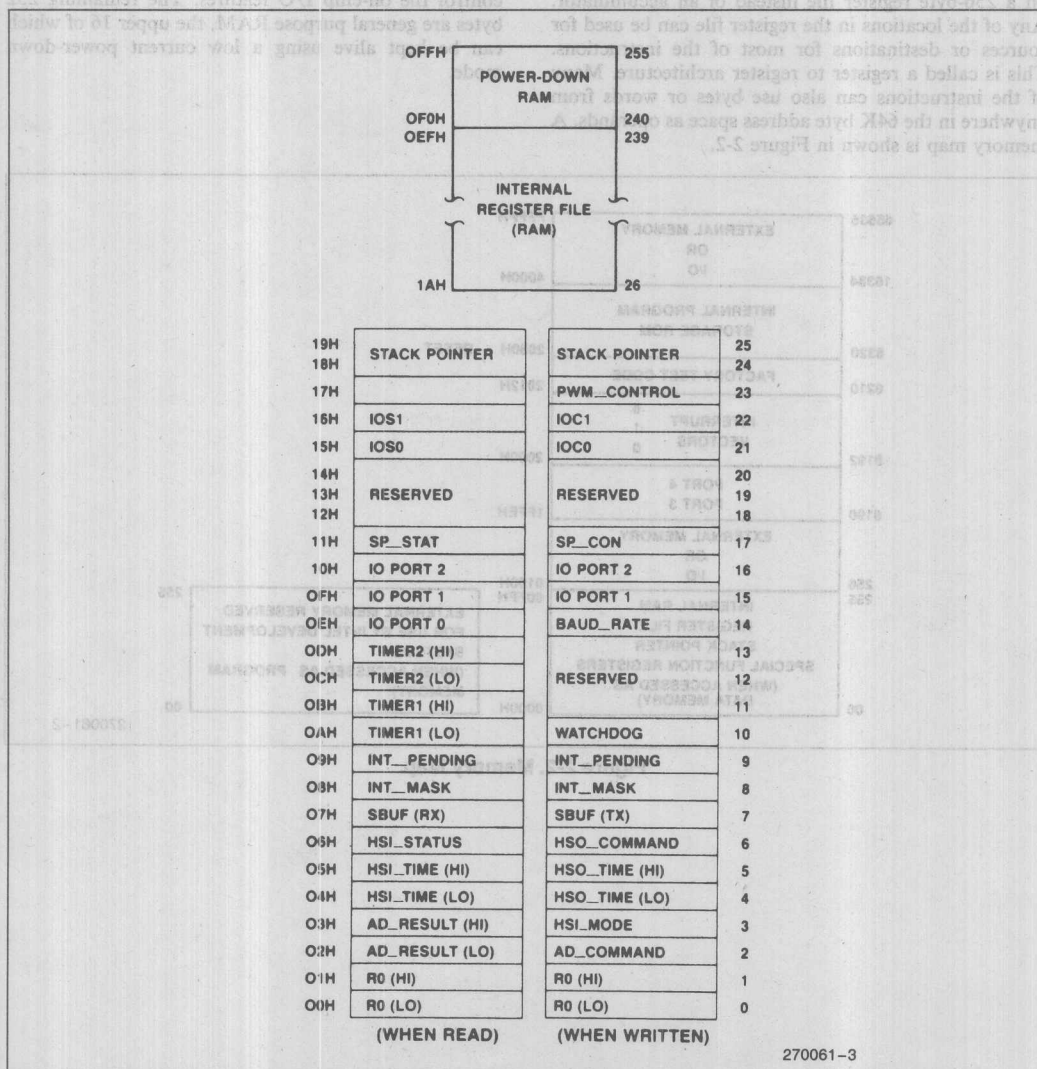


Figure 2-3: SFR Layout

2.1.2. I/O FEATURES

Many of the I/O features on the 8096 are designed to operate with little CPU intervention. A list of the major I/O functions is shown in Figure 2-4. The Watchdog Timer is an internal timer which can be used to reset the system if the software fails to operate properly. The Pulse-Width-Modulation (PWM) output can be used as a rough D to A, a motor driver, or for many other purposes. The A to D converter (ADC) has 8 multiplexed inputs and 10-bit resolution. The serial port has several modes and its own baud rate generator. The High Speed I/O section includes a 16-bit timer, a 16-bit counter, a 4-input programmable edge detector, 4 software timers, and a 6-output programmable event generator. All of these features will be described in section 2.3.

2.2. The Processor Section

2.2.1. OPERATIONS AND ADDRESSING MODES

The 8096 has 100 instructions, some of which operate on bits, some on bytes, some on words and some on longs (double words). All of the standard logical and arithmetic functions are available for both byte and word operations. Bit operations and long operations are provided for some instructions. There are also flag manipulation instructions as well as jump and call instructions. A full set of conditional jumps has been included to speed up testing for various conditions.

Bit operations are provided by the Jump Bit and Jump Not Bit instructions, as well as by immediate masking of bytes. These bit operations can be performed on any of the bytes in the register file or on any of the special function registers. The fast bit manipulation of the SFRs can provide rapid I/O operations.

A symmetric set of byte and word operations make up the majority of the 8096 instruction set. The assembly language for the 8096 (ASM-96) uses a "B" suffix on a mnemonic to indicate a byte operation, without this suffix a word operation is indicated. Many of these operations can have one, two or three operands. An example of a one operand instruction would be:

NOT Value1 ; Value1 : = 1's complement (Value1)

A two operand instruction would have the form:

ADD Value2,Value1 ; Value2 : = Value2 + Value1

A three operand instruction might look like:

MUL Value3,Value2,Value1 ;
Value3 : = Value2 * Value1

The three operand instructions combined with the register to register architecture almost eliminate the necessity of using temporary registers. This results in a faster processing time than machines that have equivalent instruction execution times, but use a standard architecture.

Long (32-bit) operations include shifts, normalize, and multiply and divide. The word divide is a 32-bit by 16-bit operation with a 16-bit quotient and 16-bit remainder. The word multiply is a word by word multiply with a long result. Both of these operations can be done in either the signed or unsigned mode. The direct unsigned modes of these instructions take only 6.5 microseconds. A normalize instruction and sticky bit flag have been included in the instruction set to provide hardware support for the software floating point package (FPAL-96).

Major I/O Functions	
High Speed Input Unit	Provides Automatic Recording of Events
High Speed Output Unit	Provides Automatic Triggering of Events and Real-Time Interrupts
Pulse Width Modulation	Output to Drive Motors or Analog Circuits
A to D Converter	Provides Analog Input
Watchdog Timer	Resets 8096 if a Malfunction Occurs
Serial Port	Provides Synchronous or Asynchronous Link
Standard I/O Lines	Provide Interface to the External World when other Special Features are not needed

Figure 2-4. Major I/O Functions

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	$D \leftarrow D + A$	✓	✓	✓	✓	↑	—	
ADD/ADDB	3	$D \leftarrow B + A$	✓	✓	✓	✓	↑	—	
ADDC/ADDCB	2	$D \leftarrow D + A + C$	↓	✓	✓	✓	↑	—	
SUB/SUBB	2	$D \leftarrow D - A$	✓	✓	✓	✓	↑	—	
SUB/SUBB	3	$D \leftarrow B - A$	✓	✓	✓	✓	↑	—	
SUBC/SUBCB	2	$D \leftarrow D - A + C - 1$	↓	✓	✓	✓	↑	—	
CMP/CMPB	2	$D - A$	✓	✓	✓	✓	↑	—	
MUL/MULU	2	$D, D + 2 \leftarrow D * A$	—	—	—	—	—	?	2
MUL/MULU	3	$D, D + 2 \leftarrow B * A$	—	—	—	—	—	?	2
MULB/MULUB	2	$D, D + 1 \leftarrow D * A$	—	—	—	—	—	?	3
MULB/MULUB	3	$D, D + 1 \leftarrow B * A$	—	—	—	—	—	?	3
DIVU	2	$D \leftarrow (D, D + 2)/A, D + 2 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	2
DIVUB	2	$D \leftarrow (D, D + 1)/A, D + 1 \leftarrow \text{remainder}$	—	—	—	✓	↑	—	3
DIV	2	$D \leftarrow (D, D + 2)/A, D + 2 \leftarrow \text{remainder}$	—	—	—	?	↑	—	2
DIVB	2	$D \leftarrow (D, D + 1)/A, D + 1 \leftarrow \text{remainder}$	—	—	—	?	↑	—	3
AND/ANDB	2	$D \leftarrow D \text{ and } A$	✓	✓	0	0	—	—	
AND/ANDB	3	$D \leftarrow B \text{ and } A$	✓	✓	0	0	—	—	
OR/ORB	2	$D \leftarrow D \text{ or } A$	✓	✓	0	0	—	—	
XOR/XORB	2	$D \leftarrow D \text{ (excl. or) } A$	✓	✓	0	0	—	—	
LD/LDB	2	$D \leftarrow A$	—	—	—	—	—	—	
ST/STB	2	$A \leftarrow D$	—	—	—	—	—	—	
LDBSE	2	$D \leftarrow A; D + 1 \leftarrow \text{SIGN}(A)$	—	—	—	—	—	—	3,4
LDBZE	2	$D \leftarrow A; D + 1 \leftarrow 0$	—	—	—	—	—	—	3,4
PUSH	1	$SP \leftarrow SP - 2; (SP) \leftarrow A$	—	—	—	—	—	—	
POP	1	$A \leftarrow (SP); SP \leftarrow SP + 2$	—	—	—	—	—	—	
PUSHF	0	$SP \leftarrow SP - 2; (SP) \leftarrow \text{PSW};$ $\text{PSW} \leftarrow 0000\text{H}$ $I \leftarrow 0$	0	0	0	0	0	0	
POPF	0	$\text{PSW} \leftarrow (SP); SP \leftarrow SP + 2; I \leftarrow$ ✓	✓	✓	✓	✓	✓	✓	
SJMP	1	$PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LJMP	1	$PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5
BR (indirect)	1	$PC \leftarrow (A)$	—	—	—	—	—	—	
SCALL	1	$SP \leftarrow SP - 2; (SP) \leftarrow PC;$ $PC \leftarrow PC + 11\text{-bit offset}$	—	—	—	—	—	—	5
LCALL	1	$SP \leftarrow SP - 2; (SP) \leftarrow PC;$ $PC \leftarrow PC + 16\text{-bit offset}$	—	—	—	—	—	—	5
RET	0	$PC \leftarrow (SP); SP \leftarrow SP + 2$	—	—	—	—	—	—	
J (conditional)	1	$PC \leftarrow PC + 8\text{-bit offset (if taken)}$	—	—	—	—	—	—	5
JC	1	Jump if C = 1	—	—	—	—	—	—	5
JNC	1	Jump if C = 0	—	—	—	—	—	—	5
JE	1	Jump if Z = 1	—	—	—	—	—	—	5

Figure 2-5. Instruction Summary

NOTES:

1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
2. D, D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D, D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to a word.
5. Offset is a 2's complement number.

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
JNE	1	Jump if Z = 0	—	—	—	—	—	—	5
JGE	1	Jump if N = 0	—	—	—	—	—	—	5
JLT	1	Jump if N = 1	—	—	—	—	—	—	5
JGT	1	Jump if N = 0 and Z = 0	—	—	—	—	—	—	5
JLE	1	Jump if N = 1 or Z = 1	—	—	—	—	—	—	5
JH	1	Jump if C = 1 and Z = 0	—	—	—	—	—	—	5
JNH	1	Jump if C = 0 or Z = 1	—	—	—	—	—	—	5
JV	1	Jump if V = 1	—	—	—	—	—	—	5
JNV	1	Jump if V = 0	—	—	—	—	—	—	5
JVT	1	Jump if VT = 1; Clear VT	—	—	—	—	0	—	5
JNVT	1	Jump if VT = 0; Clear VT	—	—	—	—	0	—	5
JST	1	Jump if ST = 1	—	—	—	—	—	—	5
JNST	1	Jump if ST = 0	—	—	—	—	—	—	5
JBS	3	Jump if Specified Bit = 1	—	—	—	—	—	—	5, 6
JBC	3	Jump if Specified Bit = 0	—	—	—	—	—	—	5, 6
DJNZ	1	D ← D - 1; if D ≠ 0 then PC ← PC + 8-bit offset	—	—	—	—	—	—	5
DEC/DECB	1	D ← D - 1	✓	✓	✓	✓	↑	—	
NEG/NEGB	1	D ← 0 - D	✓	✓	✓	✓	↑	—	
INC/INCB	1	D ← D + 1	✓	✓	✓	✓	↑	—	
EXT	1	D ← D; D + 2 ← Sign (D)	✓	✓	0	0	—	—	2
EXTB	1	D ← D; D + 1 ← Sign (D)	✓	✓	0	0	—	—	3
NOT/NOTB	1	D ← Logical Not (D)	✓	✓	0	0	—	—	
CLR/CLRB	1	D ← 0	1	0	0	0	—	—	
SHL/SHLB/SHLL	2	C ← msb ———— lsb ← 0	✓	?	✓	✓	↑	—	7
SHR/SHRB/SHRL	2	0 → msb ———— lsb → C	✓	?	✓	0	—	✓	7
SHRA/SHRAB/SHRAL	2	msb → msb ———— lsb → C	✓	✓	✓	0	—	✓	7
SETC	0	C ← 1	—	—	1	—	—	—	
CLRC	0	C ← 0	—	—	0	—	—	—	
CLRVT	0	VT ← 0	—	—	—	—	0	—	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	—	—	—	—	—	—	
EI	0	Enable All Interrupts (I ← 1)	—	—	—	—	—	—	
NOP	0	PC ← PC + 1	—	—	—	—	—	—	
SKIP	0	PC ← PC + 2	—	—	—	—	—	—	
NORML	2	Left Shift Till msb = 1; D ← shift count	✓	?	0	—	—	—	7
TRAP	0	SP ← SP - 2; (SP) ← PC PC ← (2010H)	—	—	—	—	—	—	9

Figure 2-5. Instruction Summary (Continued)

NOTES:

1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
5. Offset is a 2's complement number.
6. Specified bit is one of the 2048 bits in the register file.
7. The "L" (Long) suffix indicates double-word operation.
8. Initiates a Reset by pulling RESET low. Software should re-initialize all the necessary registers with code starting at 2080H.
9. The assembler will not accept this mnemonic.

With any one of six addressing modes. These modes increase the flexibility and overall execution speed of the 8096. The addressing modes are: register-direct, immediate, indirect, indirect with auto-increment, and long and short indexed.

The fastest instruction execution is gained by using either register direct or immediate addressing. Register-direct addressing is similar to normal direct addressing, except that only addresses in the register file or SFRs can be addressed. The indexed mode is used to directly address the remainder of the 64K address space. Immediate addressing operates as would be expected, using the data following the opcode as the operand.

Both of the indirect addressing modes use the value in a word register as the address of the operand. If the indirect auto-increment mode is used then the word register is incremented by one after a byte access or by two after a word access. This mode is particularly useful for accessing lookup tables.

Access to any of the locations in the 64K address space can be obtained by using the long indexed addressing

added to the contents of a word register to form the address of the operand. By using the zero register as the index, ASM96 (the assembler) can accept "direct" addressing to any location. The zero register is located at 0000H and always has a value of zero. A short indexed mode is also available to save some time and code. This mode uses an 8-bit 2's complement number as the offset instead of a 16-bit number.

2.2.2. ASSEMBLY LANGUAGE

The multiple addressing modes of the 8096 make it easy to program in assembly language and provide an excellent interface to high level languages. The instructions accepted by the assembler consist of mnemonics followed by either addresses or data. A list of the mnemonics and their functions are shown in Figure 2-5. The addresses or data are given in different formats depending on the addressing mode. These modes and formats are shown in Figure 2-6.

Additional information on 8096 assembly language is available in the MCS-96 Macro Assembler Users Guide, listed in the bibliography.

Mnem	Dest or Src1	: One operand direct
Mnem	Dest, Src1	: Two operand direct
Mnem	Dest, Src1, Src2	: Three operand direct
Mnem	#Src1	: One operand immediate
Mnem	Dest, #Src1	: Two operand immediate
Mnem	Dest, Src1, #Src2	: Three operand immediate
Mnem	[addr]	: One operand indirect
Mnem	[addr] +	: One operand indirect auto-increment
Mnem	Dest, [addr]	: Two operand indirect
Mnem	Dest, [addr] +	: Two operand indirect auto-increment
Mnem	Dest, Src1, [addr]	: Three operand indirect
Mnem	Dest, Src1, [addr] +	: Three operand indirect auto-increment
Mnem	Dest, offs [addr]	: Two operand indexed (short or long)
Mnem	Dest, Src1, offs [addr]	: Three operand indexed (short or long)
Where: "Mnem" is the instruction mnemonic		
"Dest" is the destination register		
"Src1", "Src2" are the source registers		
"addr" is a register containing a value to be used in computing the address of an operand		
"offs" is an offset used in computing the address of an operand		

270061-B3

Figure 2-6. Instruction Format

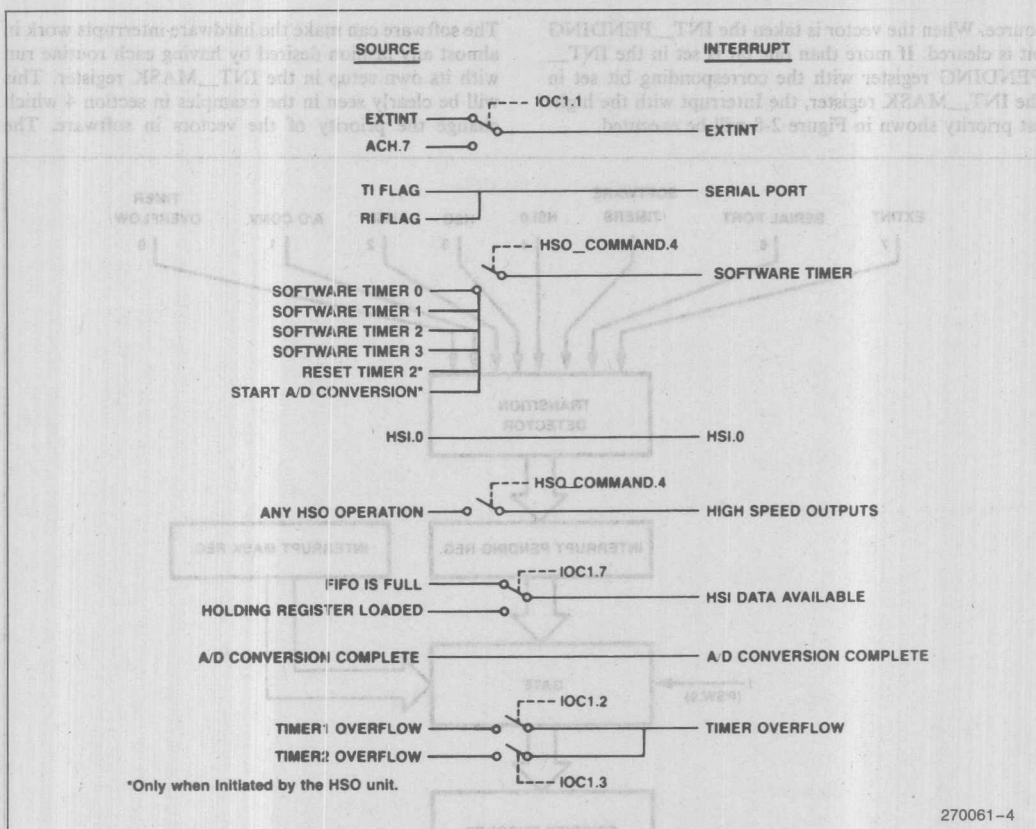


Figure 2-7. Interrupt Sources

2.2.3. INTERRUPTS

The flexibility of the instruction set is carried through into the interrupt system. There are 20 different interrupt sources that can be used on the 8096. The 20 sources vector through 8 locations or interrupt vectors. The vector names and their sources are shown in Figure 2-7, with their locations listed in Figure 2-8. Control of the interrupts is handled through the Interrupt Pending Register (INT_PENDING), the Interrupt Mask Register (INT_MASK), and the I bit in the PSW (PSW.9). Figure 2-9 shows a block diagram of the interrupt structure. The INT_PENDING register contains bits which get set by hardware when an interrupt occurs. If the interrupt mask register bit for that source is a 1 and PSW.9 = 1, a vector will be taken to the address listed in the interrupt vector table for that

Source	Vector Location		Priority
	(High Byte)	(Low Byte)	
Software	2011H	2010H	Not Applicable
Extint	200FH	200EH	7 (Highest)
Serial Port	200DH	200CH	6
Software Timers	200BH	200AH	5
HSI.0	2009H	2008H	4
High Speed Outputs	2007H	2006H	3
HSI Data Available	2005H	2004H	2
A/D Conversion Complete	2003H	2002H	1
Timer Overflow	2001H	2000H	0 (Lowest)

Figure 2-8. Interrupt Vectors and Priorities

source. When the vector is taken the INT_PENDING bit is cleared. If more than one bit is set in the INT_PENDING register with the corresponding bit set in the INT_MASK register, the Interrupt with the highest priority shown in Figure 2-8 will be executed.

The software can make the hardware interrupts work in almost any fashion desired by having each routine run with its own setup in the INT_MASK register. This will be clearly seen in the examples in section 4 which change the priority of the vectors in software. The

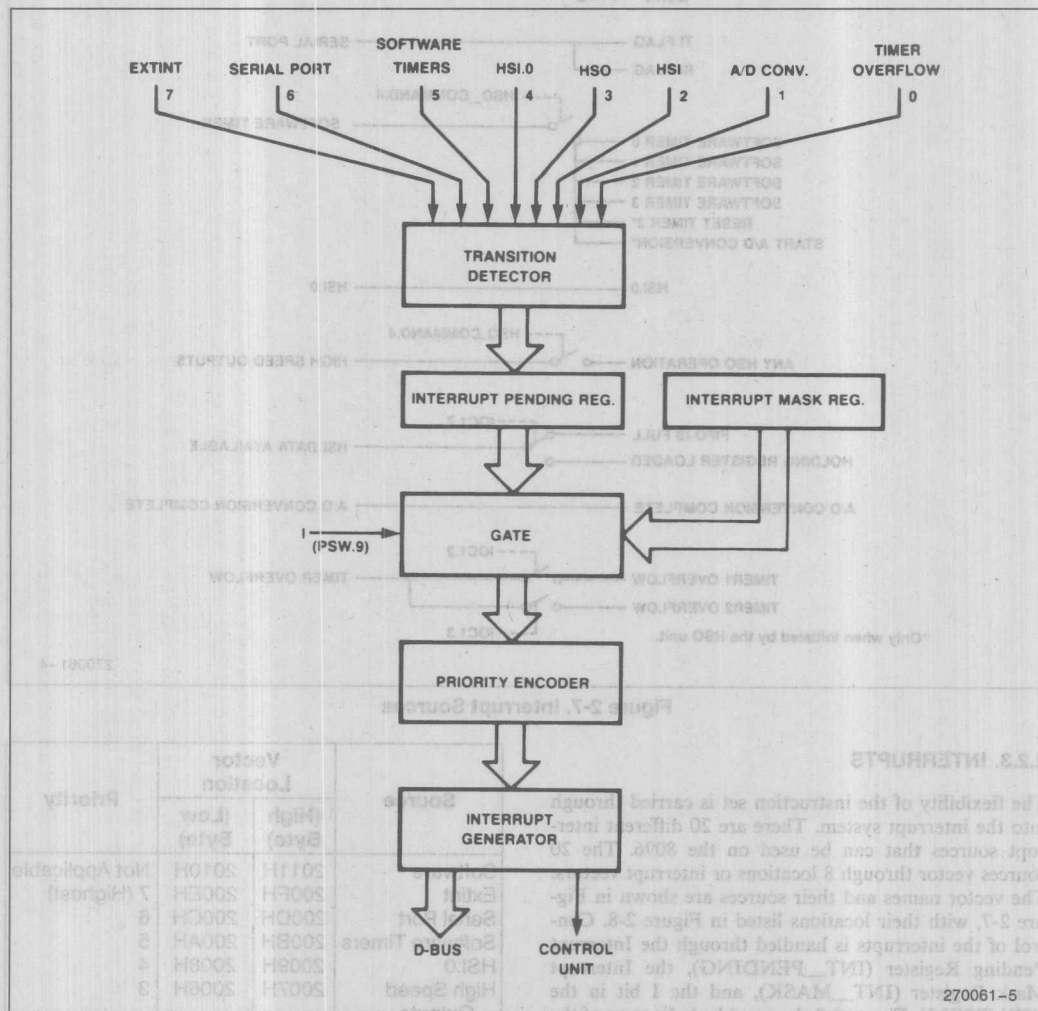


Figure 2-9. Interrupt Structure Block Diagram

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Z	N	V	VT	C	I	ST									
INT_MASK															

WHERE:

Z is the zero flag. It is set when the result of an operation is zero.
 N is the negative flag. It is set to the algebraically correct sign of the result regardless of overflows.
 V is the overflow flag. It is set if an overflow occurs.
 VT is the overflow trap flag. It is set when the VT flag is set and cleared by JVT, JNVT, or CLRVT.
 C is the carry flag. It is set if a carry was generated by the prior operation.
 I is the global interrupt enable bit.
 ST is the sticky bit. It is set during a right shift if a one was shifted into and then out of the carry flag.
 INT_MASK is the interrupt mask register and contains bits which individually enable the 8 interrupt vectors.

Figure 2-10. The PSW Register

PSW (shown in Figure 2-10), stores the INT_MASK register in its lower byte so that the mask register can be pushed and popped along with the machine status when moving in and out of routines. The action of pushing flags clears the PSW which includes PSW.9, the interrupt enable bit. Therefore, after a PUSHF instruction interrupts are disabled. In most cases an interrupt service routine will have the basic structure shown below.

INT VECTOR:

```

PUSHF
LDB INT_MASK, #xxxxxxxh
EI
;Insert service routine here
POPF
RET

```

The PUSHF instruction saves the PSW including the old INT_MASK register. The PSW, including the interrupt enable bit are left cleared. If some interrupts need to be enabled while the service routine runs, the INT_MASK is loaded with a new value and interrupts are globally enabled before the service routine continues. At the end of the service routine a POPF in-

struction is executed to restore the old PSW. The RET instruction is executed and the code returns to the desired location. Although the POPF instruction can enable the interrupts the next instruction will always execute. This prevents unnecessary building of the stack by ensuring that the RET always executes before another interrupt vector is taken.

2.3. On-Chip I/O Section

All of the on-chip I/O features of the 8096 can be accessed through the special function registers, as shown in Figure 2-3. The advantage of using register-mapped I/O is that these registers can be used as the sources or destinations of CPU operations. There are seven major I/O functions. Each one of these will be considered with a section of code to exemplify its usage. The first section covered will be the High Speed I/O, (HSIO), subsystem. This section includes the High Speed Input (HSI) unit, High Speed Output (HSO) unit, and the Timer/Counter section.

2.3.1. TIMER/COUNTERS

The 8096 has two time bases, Timer 1 and Timer 2. Timer 1 is a 16-bit free running timer which is incremented every 8 state times. (A state time is 3 oscillator periods, or 0.25 microseconds with a 12 MHz crystal.)

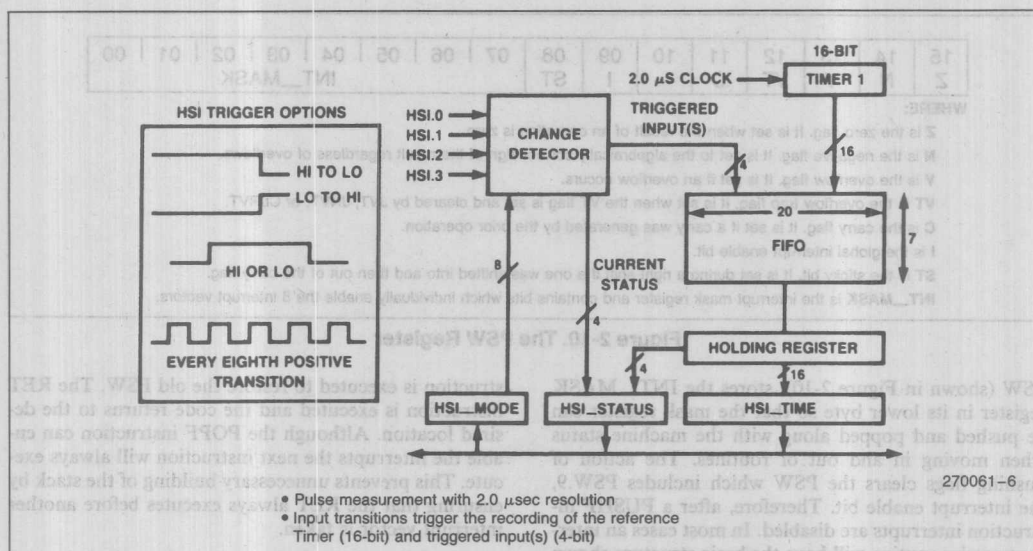


Figure 2-11. HSI Unit Block Diagram

Its value can be read at any time and used as a reference for both the HSI section and the HSO section. Timer 1 can cause an interrupt when it overflows, and cannot be modified or stopped without resetting the entire chip. Timer 2 is really an event counter since it uses an external clock source. Like Timer 1, it is 16-bits wide, can be read at any time, can be used with the HSO section, and can generate an interrupt when it overflows. Control of Timer 2 is limited to incrementing it and resetting it. Specific values can not be written to it.

Although the 8096 has only two timers, the timer flexibility is equal to a unit with many timers thanks to the HSIO unit. The HSI enables one to measure times of external events on up to four lines using Timer 1 as a timer base. The HSO unit can schedule and execute internal events and up to six external events based on the values in either Timer 1 or Timer 2. The 8096 also includes separate, dedicated timers for the baud rate generator and watchdog timer.

2.3.2. HSI

The HSI unit can be thought of as a message taker which records the line which had an event and the time at which the event occurred. Four types of events can trigger the HSI unit, as shown in the HSI block diagram in Figure 2-11. The HSI unit can measure pulse widths and record times of events with a 2

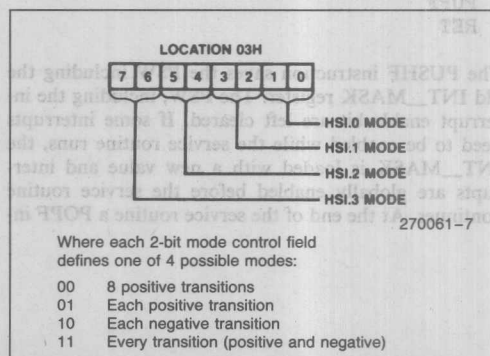


Figure 2-12. HSI Mode Register

microsecond resolution. It can look for one of four events on each of four lines simultaneously, based on the information in the HSI Mode register, shown in Figure 2-12. The information is then stored in a seven level FIFO for later retrieval. Whenever the FIFO contains information, the earliest entry is placed in the holding register. When the holding register is read, the next valid piece of information is loaded into it. Interrupts can be generated by the HSI unit at the time the

holding register is loaded or when the FIFO has six or more entries.

2.3.3. HSO

Just as the HSI can be thought of as a message taker, the HSO can be thought of as a message sender. At times determined by the software, the HSO sends mes-

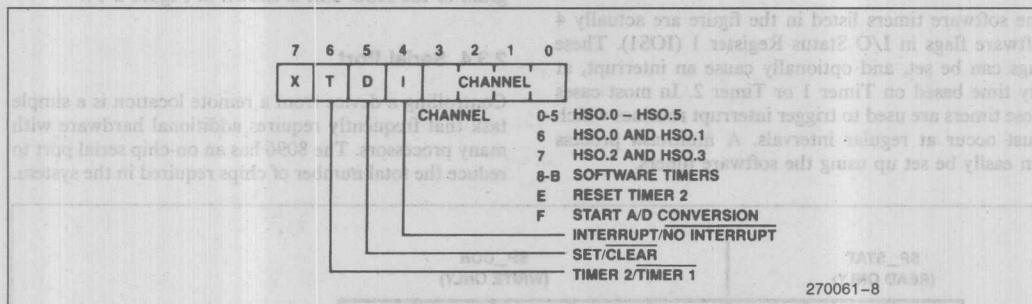


Figure 2-13. HSO Command Register

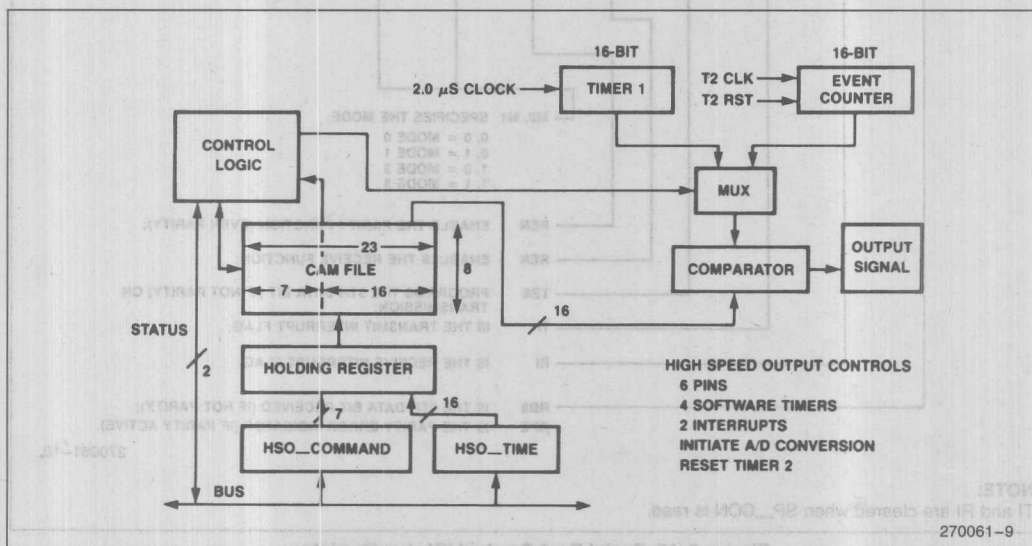


Figure 2-14. HSO Block Diagram

sages to various devices to have them turn on, turn off, start processing, or reset. Since the programmed times can be referenced to either Timer 1 or Timer 2, the HSO makes the two timers look like many. For example, if several events have to occur at specific times, the HSO unit can schedule all of the events based on a single timer. The events that can be scheduled to occur and the format of the command written to the HSO Command register are shown in Figure 2-13.

The software timers listed in the figure are actually 4 software flags in I/O Status Register 1 (IOS1). These flags can be set, and optionally cause an interrupt, at any time based on Timer 1 or Timer 2. In most cases these timers are used to trigger interrupt routines which must occur at regular intervals. A multitask process can easily be set up using the software timers.

A CAM (Content Addressable Memory) file is the main component of the HSO. This file stores up to eight events which are pending to occur. Every state time one location of the CAM is compared with the two timers. After 8 state times, (two microseconds with a 12 MHz clock), the entire CAM has been searched for time matches. If a match occurs the specified event will be triggered and that location of the CAM will be made available for another pending event. A block diagram of the HSO unit is shown in Figure 2-14.

2.3.4. Serial Port

Controlling a device from a remote location is a simple task that frequently requires additional hardware with many processors. The 8096 has an on-chip serial port to reduce the total number of chips required in the system.

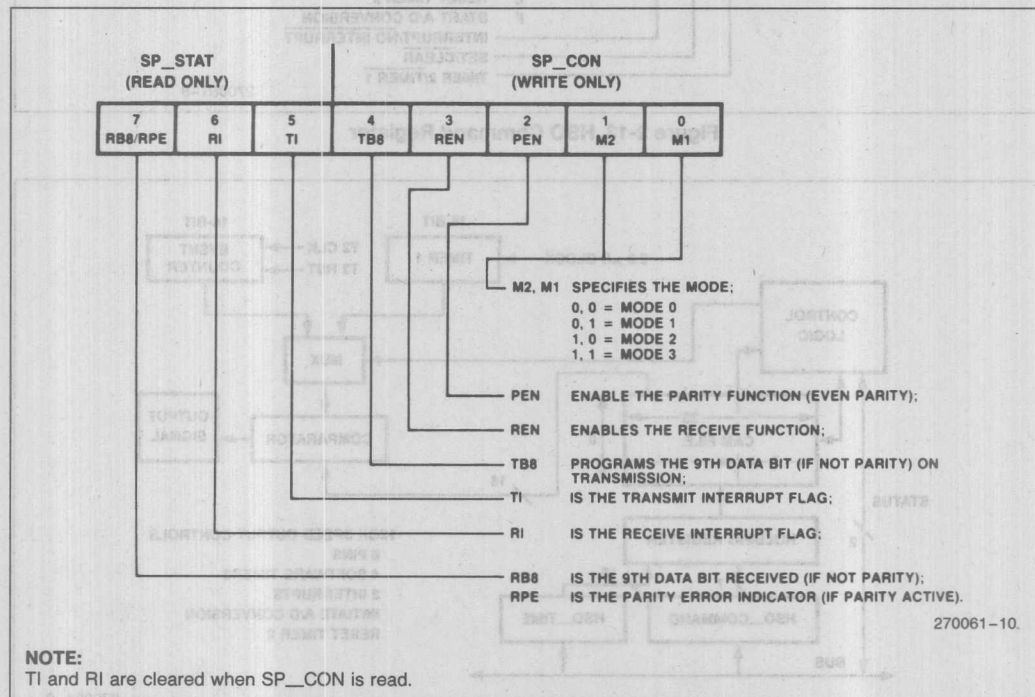


Figure 2-15. Serial Port Control/Status Register

The serial port is similar to that on the MCS-51 product line. It has one synchronous and three asynchronous modes. In the asynchronous modes baud rates of up to 187.5 Kbaud can be used, while in the synchronous mode rates up to 1.5 Mbaud are available. The chip has a baud rate generator which is independent of Timer 1 and Timer 2, so using the serial port does not take away any of the HSI, HSO or timer flexibility or functionality.

Control of the serial port is provided through the SPCON/SPSTAT (Serial Port CONTROL/Serial Port STATUS) register. This register, shown in Figure 2-15, has some bits which are read only and others which are write only. Although the functionality of the port is similar to that of the 8051, the names of some of the modes and control bits are different. The way in which the port is used from a software standpoint is also slightly different since RI and TI are cleared after each read of the register.

The four modes of the serial port are referred to as modes 0, 1, 2 and 3. Mode 0 is the synchronous mode, and is commonly used to interface to shift registers for I/O expansion. In this mode the port outputs a pulse train on the TXD pin and either transmits or receives data on the RXD pin. Mode 1 is the standard asynchronous mode, 8 bits plus a stop and start bit are sent or received. Modes 2 and 3 handle 9 bits plus a stop and start bit. The difference between the two is, that in Mode 2 the serial port interrupt will not be activated unless the ninth data bit is a one; in Mode 3 the interrupt is activated whenever a byte is received. These two modes are commonly used for interprocessor communication.

Using XTAL1:	
Mode 0: Baud Rate	$= \frac{\text{XTAL1 frequency}}{4 \cdot (B + 1)}, B \neq 0$
Others: Baud Rate	$= \frac{\text{XTAL1 frequency}}{64 \cdot (B + 1)}$
Using T2CLK:	
Mode 0: Baud Rate	$= \frac{\text{T2CLK frequency}}{B}, B \neq 0$
Others: Baud Rate	$= \frac{\text{T2CLK frequency}}{16 \cdot B}, B \neq 0$
Note that B cannot equal 0, except when using XTAL1 in other than mode 0.	

Figure 2-16. Baud Rate Formulas

Baud rates for all of the modes are controlled through the Baud Rate register. This is a byte wide register which is loaded sequentially with two bytes, and internally stores the value as a word. The least significant byte is loaded to the register followed by the most significant. The most significant bit of the baud value determines the clock source for the baud rate generator. If the bit is a one, the XTAL1 pin is used as the source, if it is a zero, the T2 CLK pin is used. The formulas shown in Figure 2-16 can be used to calculate the baud rates. The variable "B" is used to represent the least significant 15 bits of the value loaded into the baud rate register.

The baud rate register values for common baud rates are shown in Figure 2-17. These values can be used when XTAL1 is selected as the clock source for serial modes other than Mode 0. The percentage deviation from theoretical is listed to help assess the reliability of a given setup. In most cases a serial link will work if there is less than a 2.5% difference between the baud rates of the two systems. This is based on the assumption that 10 bits are transmitted per frame and the last bit of the frame must be valid for at least six-eighths of the bit time. If the two systems deviate from theoretical by 1.25% in opposite directions the maximum tolerance of 2.5% will be reached. Therefore, caution must be used when the baud rate deviation approaches 1.25% from theoretical. Note that an XTAL1 frequency of 11.0592 MHz can be used with the table values for 11 MHz to provide baud rates that have 0.0 percent deviation from theoretical. In most applications, however, the accuracy available when using an 11 MHz input frequency is sufficient.

Serial port Mode 1 is the easiest mode to use as there is little to worry about except initialization and loading and unloading SBUF, the Serial port BUFFER. If parity is enabled, (i.e., PEN=1), 7 bits plus even parity are used instead of 8 data bits. The parity calculation is done in hardware for even parity. Modes 2 and 3 are similar to Mode 1, except that the ninth bit needs to be controlled and read. It is also not possible to enable parity in Mode 2. When parity is enabled in Mode 3 the ninth bit becomes the parity bit. If parity is not enabled, (i.e., PEN = 0), the TB8 bit controls the state of the ninth transmitted bit. This bit must be set prior to each transmission. On reception, if PEN = 0, the RB8 bit indicates the state of the ninth received bit. If parity is enabled, (i.e., PEN = 1), the same bit is called RPE (Receive Parity Error), and is used to indicate a parity error.

XTAL1 Frequency = 12.0 MHz		
Baud Rate	Baud Register Value	Percent Error
19.2K	8009H	+2.40
9600	8013H	+2.40
4800	8026H	-0.16
2400	804DH	-0.16
1200	809BH	-0.16
300	8270H	0.00
XTAL1 Frequency = 11.0 MHz		
19.2K	8008H	+0.54
9600	8011H	+0.54
4800	8023H	+0.54
2400	8047H	+0.54
1200	808EH	-0.16
300	823CH	+0.01
XTAL1 Frequency = 10.0 MHz		
19.2K	8007H	-1.70
9600	800FH	-1.70
4800	8020H	+1.38
2400	8040H	-0.16
1200	8081H	-0.16
300	8208H	+0.03

Figure 2-17. Baud Rate Values for 10, 11, 12 MHz

The software used to communicate between processors is simplified by making use of Modes 2 and 3. In a basic protocol the ninth bit is called the address bit. If it is set high then the information in that byte is either the address of one of the processors on the link, or a command for all the processors. If the bit is a zero, the byte contains information for the processor or processors previously addressed. In standby mode all processors wait in Mode 2 for a byte with the address bit set. When they receive that byte, the software determines if the next message is for them. The processor that is to

receive the message switches to Mode 3 and receives the information. Since this information is sent with the ninth bit set to zero, none of the processors set to Mode 2 will be interrupted. By using this scheme the overall CPU time required for the serial port is minimized.

A typical connection diagram for the multi-processor mode is shown in Figure 2-18. This type of communication can be used to connect peripherals to a desk top computer, the axis of a multi-axis machine, or any other group of microcontrollers jointly performing a task.

Figure 2-18. Baud Rate Formulas

$$\text{Mode 0: } \text{Baud Rate} = \frac{\text{XTAL1 Frequency}}{8} \quad \text{if } \text{PEN} = 0$$

$$\text{Other: } \text{Baud Rate} = \frac{\text{XTAL1 Frequency}}{16} \quad \text{if } \text{PEN} = 1$$

Note that B cannot equal 0, except when using XTAL1 in other than mode 0.

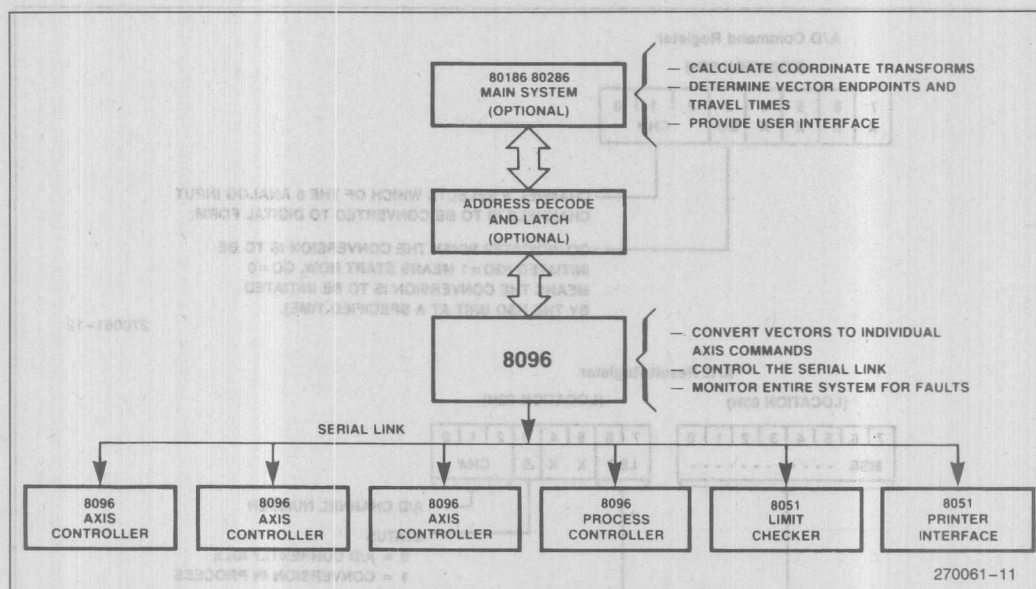


Figure 2-18. Multiprocessor Communication

Mode 0, the synchronous mode, is typically used for interfacing to shift registers for I/O expansion. The software to control this mode involves the REN (Receiver ENable) bit, the clearing of the RI bit, and writing to SBUF. To transmit to a shift register, REN is set to zero and SBUF is loaded with the information. The information will be sent and then the TI flag will be set. There are two ways to cause a reception to begin. The first is by causing a rising edge to occur on the REN bit, the second is by clearing RI with REN = 1. In either case, RI is set again when the received byte is available in SBUF.

2.3.5. A to D CONVERTER

Analog inputs are frequently required in a microcontroller application. The 8097 has a 10-bit A to D converter that can use any one of eight input channels. The conversions are done using the successive approximation method, and require 168 state times (42 microseconds with a 12 MHz clock.)

The results are guaranteed monotonic by design of the converter. This means that if the analog input voltage changes, even slightly, the digital value will either stay the same or change in the same direction as the analog

input. When doing process control algorithms, it is frequently the changes in inputs that are required, not the absolute accuracy of the value. For this reason, even if the absolute accuracy of a 10-bit converter is the same as that of an 8-bit converter, the 10-bit monotonic converter is much more useful.

Since most of the analog inputs which are monitored by a microcontroller change very slowly relative to the 42 microsecond conversion time, it is acceptable to use a capacitive filter on each input instead of a sample and hold. The 8097 does not have an internal sample and hold, so it is necessary to ensure that the input signal does not change during the conversion time. The input to the A/D must be between ANGND and VREF. ANGND must be within a few millivolts of VSS and VREF must be within a few tenths of a volt of VCC.

Using the A to D converter on the 8097 can be a very low software overhead task because of the interrupt and HSO unit structure. The A to D can be started by the HSO unit at a preset time. When the conversion is complete it is possible to generate an interrupt. By using these features the A to D can be run under complete interrupt control. The A to D can also be directly

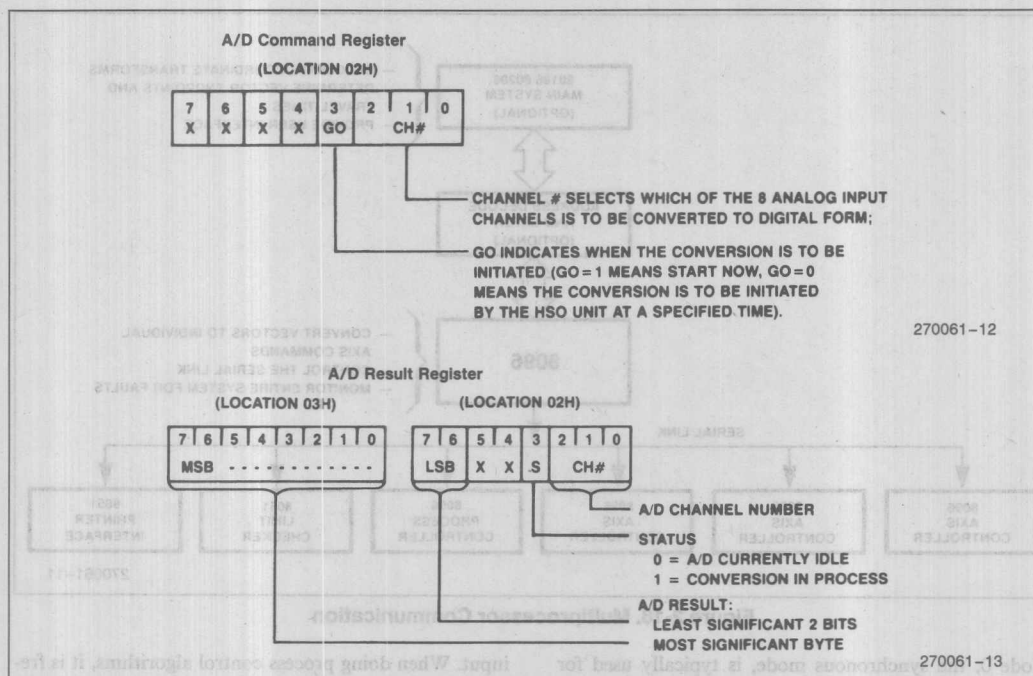


Figure 2-19. A to D Result/Command Register

controlled by software flags which are located in the AD_RESULT/AD_COMMAND Register, shown in Figure 2-19.

2.3.6. PWM REGISTER

Analog outputs are just as important as analog inputs when connecting to a piece of equipment. True digital to analog converters are difficult to make on a microprocessor because of all of the digital noise and the necessity of providing an on chip, relatively high current, rail to rail driver. They also take up a fair amount of silicon area which can be better used for other features. The A to D converter does use a D to A, but the currents involved are very small.

For many applications an analog output signal can be replaced by a Pulse Width Modulated (PWM) signal. This signal can be easily generated in hardware, and

takes up much less silicon area than a true D to A. The signal is a variable duty cycle, fixed frequency waveform that can be integrated to provide an approximation to an analog output. The frequency is fixed at a period of 64 microseconds for a 12 MHz clock speed. Controlling the PWM simply requires writing the desired duty cycle value (an 8-bit value) to the PWM Register. Some typical output waveforms that can be generated are shown in Figure 2-20.

Converting the PWM signal to an analog signal varies in difficulty, depending upon the requirements of the system. Some systems, such as motors or switching power supplies actually require a PWM signal, not a true analog one. For many other cases it is necessary only to amplify the signal so that it switches rail-to-rail, and then filter it. Switching rail-to-rail means that the output of the amplifier will be a reference value when the input is a logical one, and the output will

be zero when the input is a logical zero. The filter can be a simple RC network or an active filter. If a large amount of current is needed a buffer is also required. For low output currents, (less than 100 microamps or so), the circuit shown in Figure 2-21 can be used.

The RC network determines how quiet the output is, but the quieter the output, the slower it can change. The design of high accuracy voltage followers and active filters is beyond the scope of this paper, however many books on the subject are available.

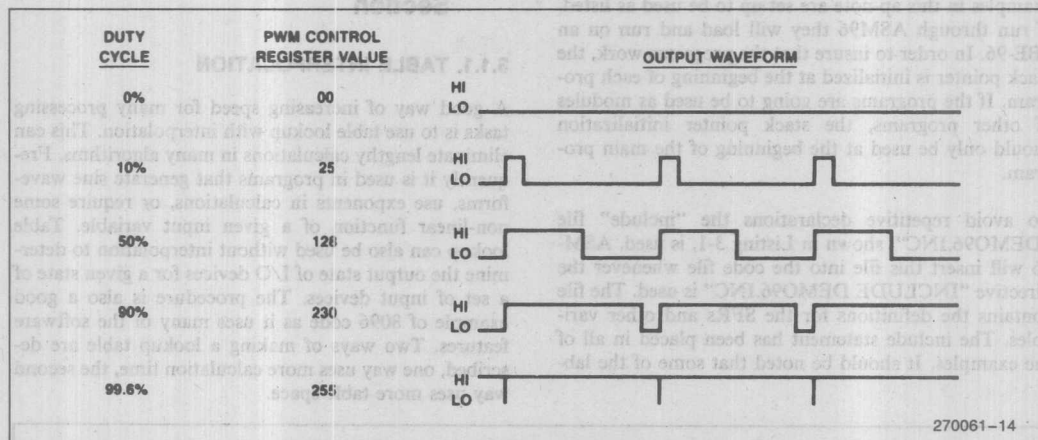


Figure 2-20. PWM Output Waveforms

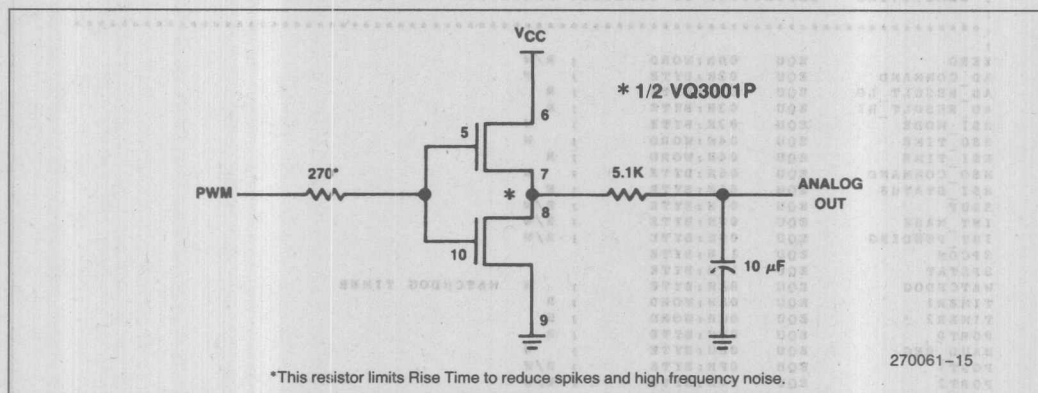


Figure 2-21. PWM to Analog Conversion Circuitry

3.0 BASIC SOFTWARE EXAMPLES

The examples in this section show how to use each I/O feature individually. Examples of using more than one feature at a time are described in section 4. All of the examples in this ap-note are set up to be used as listed. If run through ASM96 they will load and run on an SBE-96. In order to insure that the programs work, the stack pointer is initialized at the beginning of each program. If the programs are going to be used as modules of other programs, the stack pointer initialization should only be used at the beginning of the main program.

To avoid repetitive declarations the "include" file "DEMO96.INC", shown in Listing 3-1, is used. ASM-96 will insert this file into the code file whenever the directive "INCLUDE DEMO96.INC" is used. The file contains the definitions for the SFRs and other variables. The include statement has been placed in all of the examples. It should be noted that some of the lab-

els in this file are different from those in the file 8096.INC that is provided in the ASM-96 package.

3.1. Using the 8096's Processing Section

3.1.1. TABLE INTERPOLATION

A good way of increasing speed for many processing tasks is to use table lookup with interpolation. This can eliminate lengthy calculations in many algorithms. Frequently it is used in programs that generate sine waveforms, use exponents in calculations, or require some non-linear function of a given input variable. Table lookup can also be used without interpolation to determine the output state of I/O devices for a given state of a set of input devices. The procedure is also a good example of 8096 code as it uses many of the software features. Two ways of making a lookup table are described, one way uses more calculation time, the second way uses more table space.

; *****			
; DEMO96.INC - DEFINITION OF SYMBOLIC NAMES FOR THE I/O REGISTERS OF THE 8096			
; *****			
ZERO	EQU	00h:WORD	; R/W
AD_COMMAND	EQU	02h:BYTE	; R/W
AD_RESULT_LO	EQU	02h:BYTE	; R
AD_RESULT_HI	EQU	03h:BYTE	; R
HSI_MODE	EQU	03h:BYTE	; W
HSO_TIME	EQU	04h:WORD	; W
HSI_TIME	EQU	04h:WORD	; R
HSO_COMMAND	EQU	06h:BYTE	; W
HSI_STATUS	EQU	06h:BYTE	; R
SBUF	EQU	07h:BYTE	; R/W
INT_MASK	EQU	08h:BYTE	; R/W
INT_PENDING	EQU	09h:BYTE	; R/W
SPCON	EQU	11h:BYTE	
SPSTAT	EQU	11h:BYTE	
WATCHDOG	EQU	0Ah:BYTE	; W WATCHDOG TIMER
TIMER1	EQU	0Ah:WORD	; R
TIMER2	EQU	0Ch:WORD	; R
PORT0	EQU	0Eh:BYTE	; R
BAUD_REG	EQU	0Eh:BYTE	; W
PORT1	EQU	0Fh:BYTE	; R/W
PORT2	EQU	10h:BYTE	; R/W
IOC0	EQU	15h:BYTE	; W
IOS0	EQU	15h:BYTE	; R
IOC1	EQU	16h:BYTE	; W
IOS1	EQU	16h:BYTE	; R
PWM_CONTROL	EQU	17h:BYTE	; W
SP	EQU	18h:WORD	; R/W STACK POINTER
RSEG at 1CH			
AX:	DSW	1	
DX:	DSW	1	
BX:	DSW	1	
CX:	DSW	1	
AL	EQU	AX	:BYTE
AH	EQU	(AX+1)	:BYTE

270061-16

Listing 3-1. Include File DEMO.96.INC

In both methods the procedure is similar. Values of a function are stored in memory for specific input values. To compute the output function for an input that is not listed, a linear approximation is made based on the nearest inputs and nearest outputs. As an example, consider the table below.

If the input value was one of those listed then there would be no problem. Unfortunately the real world is never so kind. The input number will probably be 259 or something similar. If this is the case linear interpolation would provide a reasonable result. The formula is:

$$\text{Delta Out} = \frac{\text{Upper Output} - \text{Lower Output}}{\text{Upper Input} - \text{Lower Input}} * (\text{Actual Input} - \text{Lower Input})$$

Actual Output = Lower Output + Delta Out
For the value of 259 the solution is:

$$\text{Delta Out} = \frac{900 - 400}{300 - 200} * (259 - 200) = \frac{500}{100} * 59 = 5 * 59 = 295$$

$$\text{Actual Output} = 400 + 295 = 695$$

To make the algorithm easier, (and therefore faster), it is appropriate to limit the range and accuracy of the function to only what is needed. It is also advantageous to make the input step (Upper Input-Lower Input) equal to a power of 2. This allows the substitution of multiple right shifts for a divide operation, thus speeding up throughput. The 8096 allows multiple arithmetic right shifts with a single instruction providing a very fast divide if the divisor is a power of two.

For the purpose of an example, a program with a 12-bit output and an 8-bit input has been written. An input step of 16 ($2^{**}4$) was selected. To cover the input range 17 words are needed, $255/16 + 1$ word to handle values in the last 15 bytes of input range. Although only 12 bits are required for the output, the 16-bit architecture offers no penalty for using 16 instead of 12 bits.

The program for this example, shown in Listing 3-2, uses the definitions and equates from Listing 3-1, only the additional equates and definitions are shown in the code.

Input Value	Relative Table Address	Table Value
100	0001H	100
200	0002H	400
300	0003H	900
400	0004H	1600

```

$TITLE('INTER1.APT: Interpolation routine 1')
$INCLUDE('8096.ASM: 8096 Assembly code for table lookup and interpolation')
$INCLUDE('F1:DEMO96.INC') ; Include demo definitions

RSEG at 22H
IN_VAL:      dsb 1 ; Actual Input Value
TABLE_LOW:   dsb 1
TABLE_HIGH:  dsb 1
IN_DIF:      dsb 1 ; Upper Input - Lower Input
IN_DIFB:     equ IN_DIF, byte
TABLE_DIF:   dsb 1 ; Upper Output - Lower Output
OUT:         dsb 1
RESULT:      dsb 1
OUT_DIF:     dsb 1

CSEG at 2080H
LD SP, $100H

```

Listing 3-2. ASM-96 Code for Table Lookup Routine 1

```

look:  LDB     AL, IN_VAL      ; Load temp with Actual Value
       SHRB    AL, #3        ; Divide the byte by 8
       ANDB    AL, #1111110B ; Insure AL is a word address
                                   ; This effectively divides AL by 2
                                   ; so AL = IN_VAL/16

       LDBZ    AX, AL        ; Load byte AL to word AX
       LD      TABLE_LOW, TABLE [AX] ; TABLE_LOW is loaded with the value
                                   ; in the table at table location AX

       LD      TABLE_HIGH, (TABLE+2)[AX] ; TABLE_HIGH is loaded with the
                                   ; value in the table at table
                                   ; location AX+2
                                   ; (The next value in the table)

       SUB     TAB_DIF, TABLE_HIGH, TABLE_LOW ; TAB_DIF=TABLE_HIGH-TABLE_LOW

       ANDB    IN_DIFB, IN_VAL, #0FH ; IN_DIFB=least significant 4 bits
                                   ; of IN_VAL
       LDBZ    IN_DIF, IN_DIFB ; Load byte IN_DIFB to word IN_DIF

       MUL     OUT_DIF, IN_DIF, TAB_DIF ; Output_difference =
                                   ; Input_difference*Table_difference
                                   ; Divide by 16 (2**4)

       SHRAL   OUT_DIF, #4      ;
       ADD     OUT, OUT_DIF, TABLE_LOW ; Add output difference to output
                                   ; generated with truncated IN_VAL
                                   ; as input
       SHRA    OUT, #4          ; Round to 12-bit answer

       ADDC    OUT, zero        ; Round up if Carry = 1

no_inc: ST      OUT, RESULT      ; Store OUT to RESULT

       BR      look             ; Branch to "look;"

cseg AT 2100H
table: DCW      0000H, 2000H, 3400H, 4C00H ; A random function
       DCW      5D00H, 6A00H, 7200H, 7800H
       DCW      7B00H, 7D00H, 7600H, 6D00H
       DCW      5D00H, 4B00H, 3400H, 2200H
       DCW      1000H

END

```

270061-18

Listing 3-2. ASM-96 Code for Table Lookup Routine 1 (Continued)

If the function is known at the time of writing the software it is also possible to calculate in advance the change in the output function for a given change in the input. This method can save a divide and a few other instructions at the expense of doubling the size of the

lookup table. There are many applications where time is critical and code space is overly abundant. In these cases the code in Listing 3-3 will work to the same specifications as the previous example.

```

$TITLE('INTER2.APT: Interpolation routine 2')

; ; ; ; ; 8096 Assembly code for table lookup and interpolation
; ; ; ; ; Using tabled values in place of division

$INCLUDE('F1:DEMO96.INC') ; Include demo definitions

RSEG at 24H

IN_VAL:    dsb      1          ; Actual Input Value
TABLE_LOW: dsb      1          ; Table value for function
TABLE_INC: dsb      1          ; Incremental change in function
IN_DIF:    dsb      1          ; Upper Input - Lower Input
IN_DIFB:    equ     IN_DIF, byte
OUT:        dsb      1
RESULT:     dsb      1
OUT_DIF:    dsb      1          ; Delta Out

```

270061-19

Listing 3-3. ASM-96 Code For Table Lookup Routine 2

```

CSEG at 2080H
LD      SP, #100H      ; Initialize SP to top of reg. file
look:   LDB      AL, IN_VAL      ; Load temp with Actual Value
        SHRB     AL, #3         ; Divide the byte by 8
        ANDB     AL, $1111110B  ; This effectively divides AL by 2
        ; so AL = IN_VAL/16
        LDBZ     AX, AL         ; Load byte AL to word AX
        LD       TABLE_LOW, VAL_TABLE[AX] ; TABLE_LOW is loaded with the value
        ; in the value table at location AX
        LD       TABLE_INC, INC_TABLE[AX] ; TABLE_INC is loaded with the value
        ; in the increment table at
        ; location AX
        ANDB     IN_DIFB, IN_VAL, #0FH ; IN_DIFB=least significant 4 bits
        ; of IN_VAL
        LDBZ     IN_DIF, IN_DIFB ; Load byte IN_DIFB to word IN_DIF
        MUL      OUT_DIF, IN_DIF, TABLE_INC ; Output_difference =
        ; Input_difference*Incremental_change
        ADD      OUT, OUT_DIF, TABLE_LOW ; Add output difference to output
        ; generated with truncated IN_VAL
        ; as input
        SHR      OUT, #4        ; Round to 12-bit answer
        ADDC     OUT, zero      ; Round up if Carry = 1
no_inc: ST      OUT, RESULT      ; Store OUT to RESULT
        BR       look          ; Branch to "look:"
cseg    AT 2100H
val_table:
        DCW      0000H, 2000H, 3400H, 4C00H ; A random function
        DCW      5D00H, 6A00H, 7200H, 7800H
        DCW      7B00H, 7D00H, 7600H, 6D00H
        DCW      5D00H, 4B00H, 3400H, 2200H
        DCW      1000H
inc_table:
        DCW      0200H, 0140H, 0180H, 0110H ; Table of incremental
        DCW      00D0H, 0080H, 0060H, 0030H ; differences
        DCW      00020H, 0FF90H, 0FF70H, 0FF00H
        DCW      0FEE0H, 0FE90H, 0FEE0H, 0FEE0H
END

```

Listing 3-3. ASM-96 Code for Table Lookup Routine 2 (Continued)

By making use of the second lookup table, one word of RAM was saved and 16 state times. In most cases this time savings would not make much of a difference, but when pushing the processor to the limit, microseconds can make or break a design.

3.1.2. PL/M-96

Intel provides high level language support for most of its micro processors and microcontrollers in the form of PL/M. Specifically, PL/M refers to a family of languages, each similar in syntax, but specialized for the device for which it generates code. The PL/M syntax is similar to PL/1, and is easy to learn. PLM-96 is the version of PL/M used for the 8096. It is very code efficient as it was written specifically for the MCS-96 family. PLM-96 most closely resembles PLM-86, although it has bit and I/O functions similar to PLM-51. One line of PL/M-code can take the place of many

lines of assembly code. This is advantageous to the programmer, since code can usually be written at a set number of lines per hour, so the less lines of code that need to be written, the faster the task can be completed.

If the first example of interpolation is considered, the PLM-96 code would be written as shown in Listing 3-4. Note that version 1.0 of PLM-96 does not support 32-bit results of 16 by 16 multiplies, so the ASM-96 procedure "DMPY" is used. Procedure DMPY, shown in Listing 3-5, must be assembled and linked with the compiled PLM-96 program using RL-96, the relocater and linker. The command line to be used is:

```

RL96 PLMEX1.OBJ, DMPY.OBJ, PLM96.LIB &
to PLMOUT.OBJ ROM (2080H-3FFFH)

```



```

PLMEX:      DO;

DECLARE IN_VAL      WORD      PUBLIC;
DECLARE TABLE_LOW  INTEGER    PUBLIC;
DECLARE TABLE_HIGH INTEGER    PUBLIC;
DECLARE TABLE_DIF  INTEGER    PUBLIC;
DECLARE OUT         INTEGER    PUBLIC;
DECLARE RESULT      INTEGER    PUBLIC;
DECLARE OUT_DIF     LONGINT     PUBLIC;
DECLARE TEMP        WORD      PUBLIC;

DECLARE TABLE(17)  INTEGER DATA ( /* A random function */
0000H, 2000H, 3400H, 4C00H,
5D00H, 6A00H, 7200H, 7800H,
7B00H, 7D00H, 7600H, 6D00H,
5D00H, 4B00H, 3400H, 2200H,
1000H);

DMPY:      PROCEDURE (A,B) LONGINT EXTERNAL;
           DECLARE (A,B) INTEGER;
END DMPY;

LOOP:
  TEMP=SHR(IN_VAL,4); /* TEMP is the most significant 4 bits of IN_VAL */
  TABLE_LOW=TABLE(TEMP); /* If "TEMP" was replaced by "SHR(IN_VAL,4)" */
  TABLE_HIGH=TABLE(TEMP+1); /* The code would work but the 8096 would */
                             /* do two shifts */
  TABLE_DIF=TABLE_HIGH-TABLE_LOW;
  OUT_DIF=DMPY(TABLE_DIF,SIGNED(IN_VAL AND 0FH)) /16;
  OUT=SAR((TABLE_LOW+OUT_DIF),4); /* SAR performs an arithmetic right shift,
                                in this case 4 places are shifted */
  IF CARRY=0 THEN RESULT=OUT; /* Using the hardware flags must be done */
  ELSE RESULT=OUT+1;          /* with care to ensure the flag is tested */
                              /* in the desired instruction sequence */
GOTO LOOP;

/* END OF PLM-96 CODE */

END;

```

270061-21

Listing 3-4. PLM-96 Code For Table Lookup Routine 1

```

$TITLE('MULT.APT: 16*16 multiply procedure for PLM-96')

```

```

SP EQU 18H; word
rseq
EXTRN PLMREG; long
cseq
PUBLIC DMPY ; Multiply two integers and return a
              ; longint result in AX, DX registers
DMPY: POP PLMREG+4 ; Load return address
      POP PLMREG ; Load one operand
      MUL PLMREG,[SP] ; Load second operand and increment SP
      BR [PLMREG+4] ; Return to PLM code.
END

```

270061-22

Listing 3-5. 32-Bit Result Multiply Procedure For PLM-96

Using PLM, code requires less lines, is much faster to write, and easier to maintain, but may take slightly longer to run. For this example, the assembly code generated by the PLM-96 compiler takes 56.75 microseconds to run instead of 30.75 microseconds. If PLM-96 performed the 32-bit result multiply instead of using the ASM-96 routine the PLM code would take 41.5 microseconds to run. The actual code listings are shown in Appendix A.

3.2. Using the I/O Section

3.2.1. USING THE HSI UNIT

One of the most frequent uses of the HSI is to measure the time between events. This can be used for frequency determination in lab instruments, or speed/acceleration information when connected to pulse type encoders. The code in Listing 3-6 can be used to determine the high and low times of the signals on two lines. This code can be easily expanded to 4 lines and can also be modified to work as an interrupt routine.

Frequently it is also desired to keep track of the number of events which have occurred, as well as how often they are occurring. By using a software counter this feature can be added to the above code. This code depends on the software responding to the change in line state before the line changes again. If this cannot be guaranteed then it may be necessary to use 2 HSI lines for each incoming line. In this case one HSI line would look for falling edges while the other looks for rising edges. The code in Listing 3-7 includes both the counter feature and the edge detect feature.

The uses for this type of routine are almost endless. In instrumentation it can be used to determine frequency on input lines, or perhaps baud rate for a self adjusting serial port. Section 4.2 contains an example of making a software serial port using the HSI unit. Interfacing to some form of mechanically generated position information is a very frequent use of the HSI. The applications in this category include motor control, precise positioning (print heads, disk drives, etc.), engine control and

```

$TITLE('PULSE.APT: Measuring pulses using the HSI unit')
$INCLUDE(DEMO96.INC)

rseg at 28H
HIGH_TIME: dsw 1
LOW_TIME:  dsw 1
PERIOD:    dsw 1
HI_EDGE:   dsw 1
LO_EDGE:   dsw 1

cseg at 2080H
LD SP, #100H
LDB IOC0, #00000001B ; Enable HSI 0
LDB HSI_MODE, #00001111B ; HSI 0 look for either edge

wait: ADD PERIOD, HIGH_TIME, LOW_TIME
JBS IOS1, 6, contin ; If FIFO is full
JBC IOS1, 7, wait ; Wait while no pulse is entered

contin: LDB AL, HSI_STATUS ; Load status; Note that reading
; HSI TIME clears HSI_STATUS
LD BX, HSI_TIME ; Load the HSI_TIME
JBS AL, 1, hsi_hi ; Jump if HSI.0 is high

hsi_lo: ST BX, LO_EDGE
SUB HIGH_TIME, LO_EDGE, HI_EDGE
BR wait

hsi_hi: ST BX, HI_EDGE
SUB LOW_TIME, HI_EDGE, LO_EDGE
BR wait

END

```

Listing 3-6. Measuring Pulses Using The HSI Unit

transmission control. The HSI unit is used extensively in the example in section 4.3.

3.2.2. USING THE HSO UNIT

Although the HSO has many uses, the best example is that of a multiple PWM output. This program, shown in Listing 3-8, is simple enough to be easily understood, yet it shows how to use the HSO for a task which can be complex. In order for this program to operate, another program needs to set up the on and off time variables for each line. The program also requires that a

HSO line not change so quickly that it changes twice between consecutive reads of I/O Status Register 0, (IOS0).

A very eye catching example can be made by having the program output waveforms that vary over time. The driver routine in Listing 3-10 can be linked to the above program to provide this function. Linking is accomplished using RL96, the relocatable linker for the 8096. Information for using RL96 can be found in the "MCS-96 Utilities Users Guide", listed in the bibliography. In order for the program to link, the register dec-

```

$TITLE ('ENHNSI.APT: ENHANCED HSI_PULSE ROUTINE')
$INCLUDE (DEMO96.INC)
RSEG AT 28H

TIME: DSW 1
LAST_RISE: DSW 1
LAST_FALL: DSW 1
HSI_SO: DSB 1
IOS1_BAK: DSB 1
PERIOD: DSW 1
LOW_TIME: DSW 1
HIGH_TIME: DSW 1
COUNT: DSW 1

cseg at 2080H
init: LD SP,#100H
LDB IOC1,#00100101B ; Disable HSO.4,HSO.5, HSI_INT-first,
; Enable PWM,TXD,TIMER1_OVERFLOW_INT
LDB HSI_MODE,#10011001B ; set hsi.1 -; hsi.0 +
LDB IOC0,#00000111B ; Enable hsi 0,1
; T2 CLOCK-T2CLK, T2RST-T2RST
; Clear timer2

wait: ANDB IOS1_BAK,#01111111B ; Clear IOS1_BAK.7
ORB IOS1_BAK,IOS1 ; Store into temp to avoid clearing
; other flags which may be needed
JBC IOS1_BAK,7,wait ; If hsi is not triggered then
; jump to wait

ANDB HSI_SO,HSI_STATUS,#01010101B
LD TIME,HSI_TIME
JBS HSI_SO,0,a_rise
JBS HSI_SO,2,a_fall
BR no_cnt

a_rise: SUB LOW_TIME,TIME, LAST_FALL
SUB PERIOD,TIME, LAST_RISE
LD LAST_RISE,TIME
BR increment

a_fall: SUB HIGH_TIME,TIME, LAST_RISE
SUB PERIOD,TIME, LAST_FALL
LD LAST_FALL,TIME

increment: INC COUNT

no_cnt: BR wait

END

```

Listing 3-7. Enhanced HSI Pulse Measurement Routine

laration section (i.e., the section between "RSEG" and "CSEG") in Listing 3-8 must be changed to that in Listing 3-9.

The driver routine simply changes the duty cycle of the waveform and sets the second HSO output to a fre-

quency twice that of the first one. A slightly different driver routine could easily be the basis for a switching power supply or a variable frequency/variable voltage motor driver. The listing of the driver routine is shown in Listing 3-10.

```

; NOTE: Use this file to replace the declaration section of
; the HSO PWM program from "$INCLUDE(DEMO96.INC)" through
; the line prior to the label "wait". Also change the last
; branch in the program to a "RET".
RSEG

D_STAT: DSB 1
extrn HSO_ON_0:word, HSO_OFF_0:word
extrn HSO_ON_1:word, HSO_OFF_1:word
extrn HSO_TIME:word, HSO_COMMAND:byte
extrn TIMER1:word, IOS0:byte
extrn SP:word

public OLD_STAT
OLD_STAT: dsb 1
NEW_STAT: dsb 1

cseg
PUBLIC wait

```

270061-26

Listing 3-9. Changes to Declarations for HSO Routine

```

$TITLE('HSODRV.APT: Driver module for HSO PWM program')
HSODRV MODULE MAIN, STACKSIZE(8)

PUBLIC HSO_ON_0, HSO_OFF_0
PUBLIC HSO_ON_1, HSO_OFF_1
PUBLIC HSO_TIME, HSO_COMMAND
PUBLIC SP, TIMER1, IOS0

$INCLUDE(DEMO96.INC)

rseg at 18H
EXTRN OLD_STAT, byte

HSO_ON_0: dsb 1
HSO_OFF_0: dsb 1
HSO_ON_1: dsb 1
HSO_OFF_1: dsb 1
count: dsb 1

cseg at 2080H
EXTRN wait, entry

strt: DI
LD SP, #100H
ANDB OLD_STAT, IOS0, #0FH
XORB OLD_STAT, #0FH

initial: LD CX, #0100H

loop: LD AX, #1000H
SUB BX, AX, CX
LD AX, CX

ST AX, HSO_ON_0
ST BX, HSO_OFF_0

```

270061-27

Listing 3-10. Driver Module for HSO PWM Program

```

        SHR     AX, #1
        SHR     BX, #1
        ST     AX, HSO_ON_1
        ST     BX, HSO_OFF_1

        CALL    wait

        INC     CX
        CMP     CX, #000F00H
        BNE     loop

        BR      initial

    END

```

270061-28

Listing 3-10. Driver Module for HSO PWM Program (Continued)

Since the 8096 needs to keep track of events which of-ten repeat at set intervals it is convenient to be able to have Timer 2 act as a programmable modulo counter. There are several ways of doing this. The first is to program the HSO to reset Timer 2 when Timer 2 equals a set value. A software timer set to interrupt at Timer 2 equals zero could be used to reload the CAM. This software method takes up two locations in the CAM and does not synchronize Timer 2 to the external world.

To synchronize Timer 2 externally the T2 RST (Timer 2 ReSeT) pin can be used. In this way Timer 2 will get reset on each rising edge of T2 RST. If it is desired to have an interrupt generated and time recorded when Timer 2 gets reset, the signal for its reset can be taken from HSI.0 instead of T2RST. The HSI.0 pin has its own interrupt vector which functions independently of the HSI unit.

Another option available is to use the HSI.1 pin to clock Timer 2. By using this approach it is possible to use the HSI to measure the period of events on the input to Timer 2. If both of the HSI pins are used instead of the T2RST and T2CLK pins the HSIO unit can keep track of speed and position of the rotating device with very little software overhead. This type of setup is ideal for a system like the one shown in Figure 3-1, and similar to the one used in section 4.3.

In this system a sequence of events is required based on the position of the gear which represents any piece of rotating machinery. Timer 2 holds the count of the number of tooth edges passed since the index mark. By using HSI.1 as the input to Timer 2, instead of T2 CLK, it is possible to determine tooth count and time information through the HSI. From this information instantaneous velocity and acceleration can be calculated. Having the tooth edge count in Timer 2 means

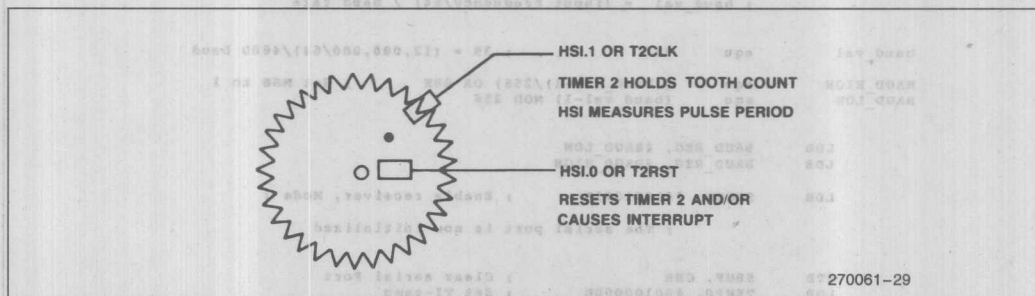


Figure 3-1. Using the HSIO to Monitor Rotating Machinery

that the HSO unit can be used to initiate the desired tasks at the appropriate tooth count. The interrupt routine initiated by HSI.0 can be used to perform any software task required every revolution. In this system, the overhead which would normally require extensive software has been done with the hardware on the 8096, thus making more software time available for control programs.

3.2.3. USING THE SERIAL PORT IN MODE 1

Mode 1 of the serial port supports the basic asynchronous 8-bit protocol and is used to interface to most CRTs and printers. The example in Listing 3-11 shows a simple routine which receives a character and then

transmits the same character. The code is set up so that minor modifications could make it run on an interrupt basis. Note that it is necessary to set up some flags as initial conditions to get the routine to run properly. If it was desired to send 7 bits of data plus parity instead of 8 bits of data the PEN bit would be set to a one. Inter-processor communication, as described in section 2.3.4, can be set up by simply adding code to change RB8 and the port mode to the listing below. The hardware shown in Figure 3-2 can be used to convert the logic level output of the 8096 to ± 12 or 15 volt levels to connect to a CRT. This circuit has been found to work with most RS-232 devices, although it does not conform to strict RS-232 specifications. If true RS-232 conformance is required then any standard RS-232 driver can be used.

<pre> \$TITLE('SP.APT: SERIAL PORT DEMO PROGRAM') \$INCLUDE(Demo96.INC) rseg at 28H CHR: ddb 1 SPTEMP: ddb 1 TEMP0: ddb 1 TEMP1: ddb 1 RCV FLAG: ddb 1 cseg at 200CH DCW ser_port_int cseg at 2080H LD SP, #100H LDB R0CL, #001000000B </pre>	
<pre> ; Baud rate = input frequency / (64*baud_val) ; baud_val = (input frequency/64) / baud_rate </pre>	
<pre> baud_val equ 39 ; 39 = (12,000,000/64)/4800 baud BAUD_HIGH equ ((baud_val-1)/256) OR 80H ; Set MSB to 1 BAUD_LOW equ (baud_val-1) MOD 256 </pre>	
<pre> LDB BAUD_REG, #BAUD_LOW LDB BAUD_REG, #BAUD_HIGH LDB SPCON, #01001001B ; Enable receiver, Mode 1 ; The serial port is now initialized </pre>	
<pre> STB SBUF, CHR ; Clear serial Port LDB TEMP0, #001000000B ; Set TI-temp </pre>	
<pre> LDB INT_MASK, #01000000B ; Enable Serial Port Interrupt EI loop: BR loop ; Wait for serial port interrupt </pre>	
<pre> ser_port_int: PUSHF rd_again: LDB SPTEMP, SPSTAT ; This section of code can be replaced ORR TEMP0, SPTEMP ; with "ORR TEMP0, SP_STAT" when the ANDB SPTEMP, #011000000B ; serial port TI and RI bugs are fixed JNE rd_again ; Repeat until TI and RI are properly cleared </pre>	

270061-30

Listing 3-11. Using the Serial Port in Mode 1

STAY: GOING THE A TO B

n does not

De

order, Port 1 output pins are used to indicate the current status of each task. The actual code listing is included in Appendix B.

The initialization section, shown in Listing 4-1a, clears a few variables and then loads the first set of on and off times to the HSO unit. Note that 8 state times must

be waited between consecutive loads of the HSO. If this is not done it is possible to overwrite the contents of the CAM holding register. An A/D interrupt is forced by setting the bit in the Interrupt Pending register. This causes the first A/D interrupt to occur just after the Interrupt Mask register is set and interrupts are enabled.

Listing 4-1. Using Multiple I/O Devices

```

; This program will provide 3 PWM outputs on HSO pins 0-2
; and one on the PWM.
;
; The PWM values are determined by the input to the A/D converter.
;
;/////////////////////////////////////////////////////////////////
$INCLUDE(DENO96.INC)
RSEG AT 20H

DL      EQU      DX:BYTE

ON_TIME:
    PWM_TIME 1:    DSW      1
    HSO_ON_0:      DSW      1
    HSO_ON_1:      DSW      1
    HSO_ON_2:      DSW      1

RESULT_TABLE:
    RESULT_0:      DSW      1
    RESULT_1:      DSW      1
    RESULT_2:      DSW      1
    RESULT_3:      DSW      1

NXT_ON_T:         DSW      1
NXT_OFF_0:        DSW      1
NXT_OFF_1:        DSW      1
NXT_OFF_2:        DSW      1
COUNT:          DSL      1
AD_NUM:           DSW      1
TMP:              DSW      1
HSO_PER:          DSW      1
LAST_LOAD:        DSB      1

cseg      AT 2000H
    DCW      start      , Timer_ovf_int
    DCW      Atod_done_int , HSI_data_int
    DCW      start      , HSI_data_int
    DCW      HSO_exec_int

cseg      AT 2080H
start:    LD      SP, #100H      , Set Stack Pointer
          CLR     AX
wait:     DEC     AX
          JNE     wait          , wait approx. 0.2 seconds for
                                SBE to finish communications
          CLRB    AD_NUM

          LD      PWM_TIME 1, #080H
          LD      HSO_PER, #100H
          LD      HSO_ON_0, #040H
          LD      HSO_ON_1, #080H
          LD      HSO_ON_2, #0C0H

          ADD     NXT_ON_T, Timer1, #100H

```

Listing 4-1a. Initializing the A to D to PWM Program

```

LDB HSO_COMMAND, #0011010B ; Set HSO for timer1, set pin 0,1
LD HSO_TIME, NXT_ON_T ; with interrupt
NOP
NOP
LDB HSO_COMMAND, #00100010B ; Set HSO for timer1, set pin 2
ADD HSO_TIME, NXT_ON_T ; without interrupt
ORB LAST_LOAD, #00000111B ; Last loaded value was set all pins
LDB INT_MASK, #00001010B ; Enable HSO and A/D interrupts
LDB INT_PENDING, #000001010B ; Fake an A/D and HSO interrupt
EI

loop: ORB Port1, #00000001B ; set P1.0
      ADD COUNT, #01
      ADDC COUNT+2, zero
      ANDB Port1, #11111101B ; clear P1.0
      BR loop

```

270061-35

Listing 4-1a. Initializing the A to D to PWM program (Continued)

```

; HSO EXECUTED INTERRUPT
;
HSO_exec_int:
PUSHF
ORB Port1, #00000010B ; Set p1.1

SUB TMP, TIMER1, NXT_ON_T
CMP TMP, ZERO
JLT set_off_times

set_on_times:
ADD NXT_ON_T, HSO_PER
LDB HSO_COMMAND, #0011010B ; Set HSO for timer1, set pin 0,1
LD HSO_TIME, NXT_ON_T
NOP
NOP
LDB HSO_COMMAND, #00100010B ; Set HSO for timer1, set pin 2
LD HSO_TIME, NXT_ON_T
ORB LAST_LOAD, #00000111B ; Last loaded value was all ones
LDB PWM_CONTROL, PWM_TIME_1 ; Now is as good a time as any
BR check_done ; to update the PWM reg

set_off_times:
JBC LAST_LOAD, 0, check_done

ADD NXT_OFF_0, NXT_ON_T, HSO_ON_0
LDB HSO_COMMAND, #00010000B ; Set HSO for timer1, clear pin 0
LD HSO_TIME, NXT_OFF_0

NOP
ADD NXT_OFF_1, NXT_ON_T, HSO_ON_1
LDB HSO_COMMAND, #00010001B ; Set HSO for timer1, clear pin 1
LD HSO_TIME, NXT_OFF_1

NOP
ADD NXT_OFF_2, NXT_ON_T, HSO_ON_2
LDB HSO_COMMAND, #00010010B ; Set HSO for timer1, clear pin 2
LD HSO_TIME, NXT_OFF_2

ANDB LAST_LOAD, #11111000B ; Last loaded value was all 0s

check_done:
ANDB Port1, #11111101B ; Clear P1.1
POPF
RET

```

270061-36

Listing 4-1b. Interrupt Driven HSO Routine

```

; A TO D COMPLETE INTERRUPT
;
; ATOD_done Int:
; PUSHF
; ORB Port1, #00000100B ; Set Pl.2
;
; ANDB AL, AD_RESULT_LO, #11000000B ; Load low order result
; LDB AH, AD_RESULT_HI ; Load high order result
; ADDB DL, AD_NUM, AD_NUM ; DL= AD_NUM *2
; LDBZ DX, DL
; ST AX, RESULT_TABLE[DX] ; Store result indexed by DX
;
; CMPB AL, #01000000B
; JNH no_rnd ; Round up if needed
; CMPB AH, #0FFH ; Don't increment if AH=0FFH
; JE no_rnd
; INCB AH
;
; no_rnd: LDB AL, AH ; Align byte and change to word
; CLRB AH
; ST AX, ON_TIME[DX]
;
; INCB AD_NUM
; ANDB AD_NUM, #03H ; Keep AD_NUM between 0 and 3
;
; next: ADDB AD_COMMAND, AD_NUM, #1000B ; Start conversion on channel
; ; indicated by AD_NUM register
; ANDB Port1, #11111011B ; Clear Pl.2
; POPF
; RET
;
; END

```

270061-37

Listing 4-1c. Interrupt Driven A to D Routine

The HSO routine shown in Listing 4-1b is slightly different than the one in section 3. All of the HSO lines turn on at the same time, only the turn-off-time is varied between lines. This action is what is most commonly required for multiple PWM outputs and simplifies the software. A comparison is made between Timer1 and the next HSO turn on time at the beginning of the routine. If the next turn on time has passed, then the on-times are loaded into the CAM, otherwise the off times are loaded.

The maximum number of events in the CAM at any given time is 7. This occurs when the first line to turn off does so, causing the off-times for all of the lines to be loaded. For two of the lines there will be an offtime, an on-time, and the just loaded off-time. The other line (the one that just turned off) will have only the on-time and the just loaded off-time.

A/D conversions are performed by the code in Listing 4-1c about every 60 microseconds, 42 for the conversion, the rest for overhead. The A/D routine sets up the HSO and PWM on and off times. Since the A/D

has a ten bit output, the most significant 8 bits are rounded up or down based on the least significant two bits.

4.2. Software Serial Port Using the HSIO Unit

There are many systems which require more than one serial port, an example is a system which must communicate with other computers and have an additional port for a local console. If the on-board UART is being used as an inter-processor link, the HSIO unit can be used to interface the 8096 to an additional asynchronous line.

Figure 4-1 shows the format of a standard 10-bit asynchronous frame. The start bit is used to synchronize the receiver to the transmitter; at the leading edge of the START bit the receiver must set up its timing logic to sample the incoming line in the center of each bit. Following the start bit are the eight data bits which are transmitted least significant bit first. The STOP bit is set to the opposite state of the START bit to guar-

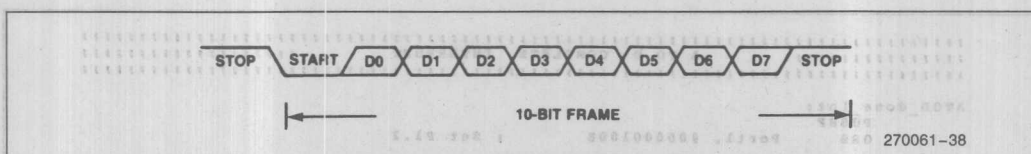


Figure 4-1. 10-bit Asynchronous Frame

antee that the leading edge of the START bit will cause a transition on the line; it also provides for a dead time on the line so that the receiver can maintain its synchronization.

The remainder of this section will show how a full-duplex asynchronous port can be built from the HSIO unit. There are four sections to this code:

1. Interface routines. These routines provide a procedural interface between the interrupt driven core of the software serial port and the remainder of the application software.
2. Initialization routine. This routine is called during the initialization of the overall system and sets up the various variables used by the software port.

3. Transmit ISR. This routine runs as an ISR (interrupt service routine) in response to an HSO interrupt interrupt. Its function is to serialize the data passed to it by the interface routines.

4. Receive ISRs. There are two ISRs involved in the receive process. One of them runs in response to an HSI interrupt and is used to synchronize the receive process at the leading edge of the start bit. The second receive ISR runs in response to an HSO generated software timer interrupt, this routine is scheduled to run at the center of each bit and is used to deserialize the incoming data.

The routines share the set of variables that are shown in Listing 4-2. These variables should be accessed only by the routines which make up the software serial port.

```

;
; VARIABLES NEEDED BY THE SOFTWARE SERIAL PORT
;
;-----
; rseg
;
; rcve_state:      dsb 1
; rxldy           equ 1      ; indicates receive done
; rxoverrun       equ 2      ; indicates receive overflow
; rip            equ 4      ; receive in progress flag
; rcve_buf:       dsb 1      ; used to double buffer receive data
; rcve_reg:       dsb 1      ; used to deserialize receive data
; sample_time:    dsb 1      ; records last receive sample time
;
; serial_out:     dsb 1      ; Holds the output character+framing (start and
; baud_count:     dsb 1      ; stop bits) for transmit process.
; txld_time:      dsb 1      ; Holds the period of one bit in units
; char:           dsb 1      ; of T1 ticks.
; txld_time:      dsb 1      ; Transition time of last Txd bit that was
; char:           dsb 1      ; sent to the CAM
;
; COMMANDS ISSUED TO THE HSO UNIT
;
; mark_command    equ 0110101b ; timer1,set,interrupt on 5
; space_command   equ 0010101b ; timer1,clr,interrupt on 5
; sample_command  equ 0011000b ; software timer 0
;
; select

```

270061-39

Listing 4-2. Software Serial Port Declarations

The table also shows the declarations for the commands issued to the HSO unit. In this example HSI.2 is used for receive data and HSO.5 is used for transmit data, although other HSI and HSO lines could have been used.

The interface routines are shown in Listing 4-3. Data is passed to the port by pushing the eight-bit character into the stack and calling `char_out`, which waits for any in-process transmission to complete and stores the character into the variable `serial_out`. As the data is

stored the START and STOP bits are added to the data bits. The routine `char_in` is called when the application software requires a character from the port. The data is returned in the `ax` register in conformance to PLM 96 calling conventions. The routine `csts` can be called to determine if a character is available at the port before calling `char_in`. (If no character is available `char_in` will wait indefinitely).

The initialization routine is shown in Listing 4-4. This routine is called with the required baud rate in the

```

; char_out:
; Output character to the software serial port
;
;   pop     cx          ; the return address
;   pop     bx          ; the character for output
;   ldb     (bx+1),%01h ; add the start and stop bits
;   add     bx,bx        ; to the char and leave as 16 bit
;
; wait_for_xmit:
;   cmp     serial_out,0 ; wait for serial_out=0 (it will be cleared by
;   bne     wait_for_xmit ; the hso interrupt process)
;   st      bx,serial_out ; put the formatted character in serial_out
;   br      [cx]         ; return to caller
;
; csts:
; Returns "true" (ax<>0) if char_in has a character.
;
;   clr     ax
;   bbc     rcve_state,0,csts_exit
;   inc     ax
; csts_exit:
;   ret
;
; char_in:
; Get a character from the software serial port
;
;   ; wait for character ready
;   bbc     rcve_state,0,char_in
;   pushf
;   andb    rcve_state,%not(rxrdy)
;   ldbz    al,rcve_buf
;   popf
;   ret

```

270061-40

Listing 4-3. Software Serial Port Interface Routines

```

; setup_serial_port:
; Called on system reset to intiate the software serial port.
;
;   pop     cx          ; the return address
;   pop     bx          ; the baud rate (in decimal)
;   ld      dx,%0007h   ; dx:ax:=500,000 (assumes 12 Mhz crystal)
;   ld      ax,%0A120h
;   divu    ax,bx        ; calculate the baud count (500,000/baudrate)
;   st      ax,baud_count
;   st      0,serial_out ; Clear serial_out
;   ldb     ioc1,%01100000b ; Enable HSO.5 and Txd
;   bbs     ioc0,6,%     ; Wait for room in the HSO CAM
;   ; and issue a MARK command.
;   add     txd_time,timer1,20
;   ldb     hso_command,%mark_command
;   ld      hso_time,txd_time
;   clrb    rcve_buf
;   clrb    rcve_reg
;   clrb    rcve_state
;   call    init_receive ; setup to detect a start bit
;   br      [cx]         ; return

```

270061-41

Listing 4-4. Software Serial Port Initialization Routine

stack; it calculates the bit time from the baud rate and stores it in the variable *baud_count* in units of TIMER1 ticks. An HSO command is issued which will initiate the transmit process and then the remainder of the variables owned by the port are initialized. The routine *init_receive* is called to setup the HSI unit to look for the leading edge of the START bit.

The transmit process is shown in Listing 4-5. The HSO unit is used to generate an output command to the transmit pin once per bit time. If the *serial_out* register is zero a MARK (idle condition) is output. If the *serial_out* register contains data then the least sig-

nificant bit is output and the register shifted right one place. The framing information (START and STOP bits) are appended to the actual data by the interface routines. Note that this routine will be executed once per bit time whether or not data is being transmitted. It would be possible to use this routine for additional low resolution timing functions with minimal overhead.

The receive process consists of an initialization routine and two interrupt service routines, *hsi_isr* and *software_timer_isr*. The listings of these routines are shown in Listings 4-6a, 4-6b, and 4-6c respectively. The

```

;
; hso_isr:
; Fields the hso interrupts and performs the serialization of the data.
; Note: this routine would be incorporated into the hso service strategy for an
; actual system.
;
; at 2006h
; hso_isr ; Set up vector
;
; cseg
; pushf
; add txd_time,baud_count
; cmp serial_out,0 ; if character is done send a mark
; be send_mark
; shr serial_out,$1 ; else send bit 0 of serial_out and shift
; bc send_mark ; serial_out left one place.
;
; send_space:
; ldb hso_command,$space_command
; ld hso_time,txd_time
; br hso_isr_exit
;
; send_mark:
; ldb hso_command,$mark_command
; ld hso_time,txd_time
;
; hso_isr_exit:
; popf
; ret
;
; select

```

270061-42

Listing 4-5. Software Serial Port Transmit Process

Listing 4-6. Receive Process

```

;
; init_receive:
; Called to prepare the serial input process to find the leading edge of
; a start bit.
;
; ldb loc0,$00000000b ; disconnect change detector
; ldb hsi_mode,$00100000b ; negative edges on HSI.2
;
; flush_fifo:
; orb iosl_save,iosl
; bbc iosl_save,7,flush_fifo_done
; ldb al,hsi_status
; ld at,hsi_time ; trash the fifo entry
; andb iosl_save,$not(80h) ; clear bit 7.
; br flush_fifo
;
; flush_fifo_done:
; ldb loc0,$00010000b ; connect HSI.2 to detector
; ret

```

270061-43

Listing 4-6a. Software Serial Port Receive Initialization

```

; hsi_isr: interrupts from the HSI unit, used to detect the leading edge
; of the START bit
; Note: this routine would be incorporated into the HSI strategy of an actual
; system.

cseg at 2004h
dcw hsi_isr ; setup the interrupt vector

cseg
pushf
push ax
ldb al, hsi_status
ld sample_time, hsi_time
bbc al, 4, exit_hsi
bbs ios0, 7, $ ; wait for room in HSO holding reg
ld ax, baud_count ; send out sample command in 1/2
shr ax, 1 ; bit time
add sample_time, ax
ldb hso_command, $sample_command
st sample_time, hso_time ; disconnect hsi.2 from change detector
ldb ios0, $00000000b

exit_hsi:
pop ax
popf
ret

```

270061-44

Listing 4-6b. Software Serial Port Start Bit Detect

```

; software_timer_isr:
; Fields: the software timer interrupt, used to deserialize the incoming data.
; Note: this routine would be incorporated into the software timer strategy
; in an actual system.

cseg at 2004h
dcw software_timer_isr ; setup vector

cseg
pushf
orb ios1_save, ios1
andb ios1_save, $not(01h) ; clear bit 0
andb 0, rcv_state, $0fch ; All bits except rxrxdy and overrun=0
bne process_data

process_start_bit:
bbc hsi_status, 5, start_ok
call init_receive
br software_timer_exit

start_ok:
orb rcv_state, $rip ; set receive in progress flag
br schedule_sample

process_data:
bbs rcv_state, 7, check_stopbit
shrb rcv_reg, $1
bbc hsi_status, 5, datazero
rcv_reg, $80h ; set the new data bit

datazero:
addb rcv_state, $10h ; increment bit count
br schedule_sample

check_stopbit:
bbc hsi_status, 5, $ ; DEBUG ONLY
ldb rcv_buf, rcv_reg
orb rcv_state, $rxrxdy
andb rcv_state, $03h ; Clear all but ready and overrun bits
call init_receive
br software_timer_exit

schedule_sample:
bbs ios0, 7, $ ; wait for holding reg empty
ldb hso_command, $sample_command
add sample_time, baud_count
st sample_time, hso_time

software_timer_exit:
popf
ret

```

270061-45

Listing 4-6c. Software Serial Port Data Reception

start is detected by the *hsi_isr* which schedules a software timer interrupt in one-half of a bit time. This first sample is used to verify that the START bit has not ended prematurely (a protection against a noisy line). The software timer service routine uses the variable *rcve_state* to determine whether it should check for a valid START bit, deserialize data, or check for a valid STOP bit. When a complete character has been received it is moved to the receive buffer and *init_receive* is called to set up the receive process for the next character. This routine is also called when an error (e.g., invalid START bit) is detected.

Appendix C contains the complete listing of the routines and the simple loop which was used to initialize them and verify their operation. The test was run for several hours at 9600 baud with no apparent malfunction of the port.

4.3. Interfacing an Optical Encoder to the HSI Unit

Optical encoders are among one of the more popular devices used to determine position of rotating equipment. These devices output two pulse trains with edges that occur from 2 to 4000 times a revolution.

Frequently there is a third line which generates one pulse per revolution for indexing purposes. Figure 4-2 shows a six line encoder and typical waveforms. As can be seen, the two waveforms provide the ability to determine both position and direction. Since a microcontroller can perform real time calculations it is possible to determine velocity and acceleration from the position and time information.

Interfacing to the encoder can be an interesting problem, as it requires connecting mechanically generated electrical signals to the HSI unit. The problems arise because it is difficult to obtain the exact nature of the signals under all conditions.

The equipment used in the lab was a Pittman 9400 series gearmotor with a 600 line optical encoder from Vernitech. The encoder has to be carefully attached to the shaft to minimize any runout or endplay. Fortunately, Pitmann has started marketing their motors with ball bearings and optical encoders already installed. It is recommended that the encoder be mounted to the motor using the exact specifications of the encoder manufacturer and/or a good machine shop.

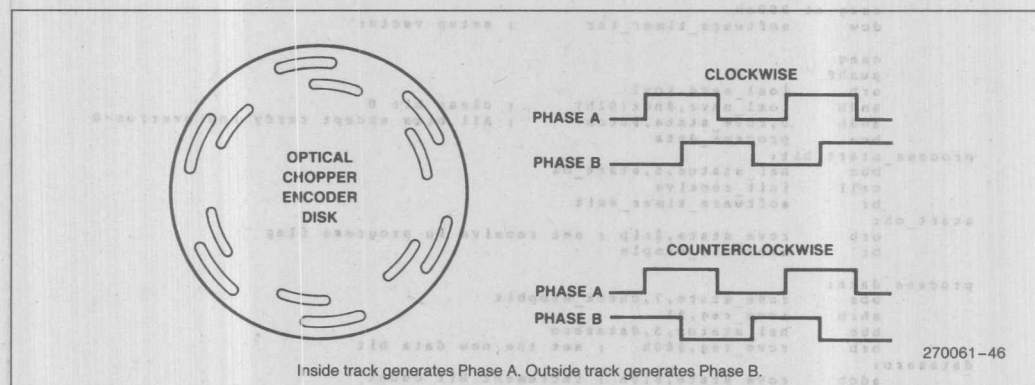


Figure 4-2. Optical Encoder and Waveforms

coder signals. The idealized signals coming from the encoder and after the digital filter are shown in Figure 4-3. The circuitry connecting the encoder to the 8096 requires only two chips. A one-shot constructed of XOR gates generates pulses on each edge of each signal. The pulses generated by Phase A are used to clock the signal from Phase B and vice versa. The hardware is shown in Figure 4-4. CMOS parts are used to reduce loading on the encoder so that buffers are not needed. Note that T2CLK is clocked on both edges of both filtered phases.

By using this method repetitive edges on a single phase without an edge on the other phase will not be passed on to the 8096. Repetitive edges on a phase can occur when the motor is stopped and vibrates or when it is changing direction. The digital filtering technique causes a little more delay in the signal at slow speeds than an analog filter would, but the simplicity trade off is worthwhile. The net effect of digital filtering is losing the ability to determine the first edge after a direction change. This does not affect the count since the first edge in both directions is lost.

fore filtering, the encoder outputs can be attached directly to the 8096. As these would be input signals, Port 0 is the most likely choice for connection. It would not be required to connect these lines to the HSI unit, as the information on them would only be needed when the motor is going very slowly.

The motor is driven using the PWM output pin for power control and a port pin for direction control. The 8096 drives a 7438 which drives 2 opto-isolators. These in turn drive two VFETs. A MOV (Metal Oxide Varistor, a type of transient absorber) is used to protect the VFETs, and a capacitor filters the PWM to get the best motor performance. Figure 4-5 shows the driver circuitry. To avoid noise getting into the 8096 system, the ± 15 volt power supply is isolated from the 8096 logic power supply.

This is the extent of the external circuitry required for this example. All of the counting and direction detection are done by the 8096. There are two sections to the example: driving the motor and interfacing to the encoder. The motor driver uses proportional control with

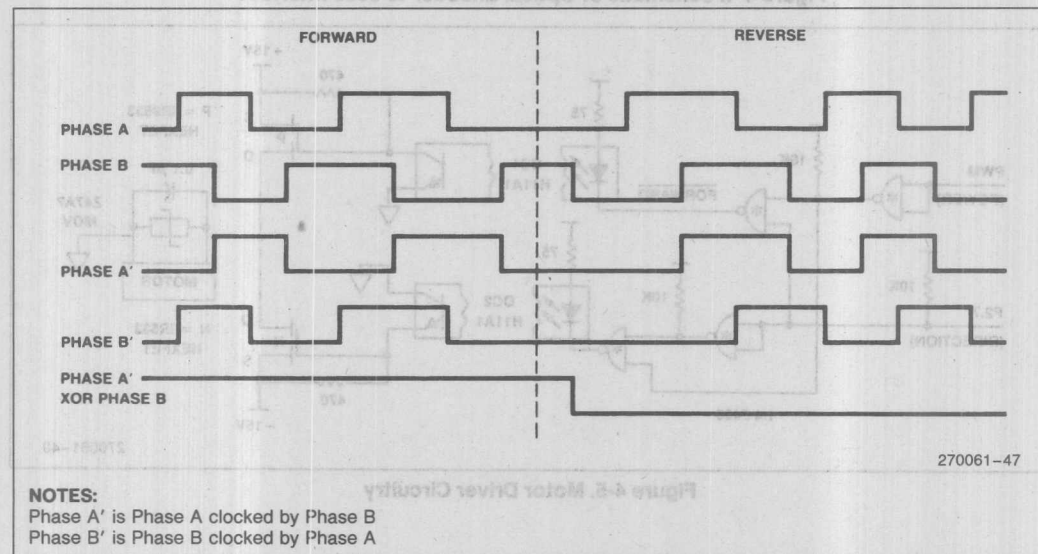


Figure 4-3. Filtered Encoder Waveforms

some modifications and a braking algorithm. Since the main point of this example is I/O interfacing, the motor driver will be briefly described at the end of this section.

In order to interface to the encoder it is necessary to know the types of waveforms that can be expected. The motor was accelerated and decelerated many times using different maximum voltages. It was found that the

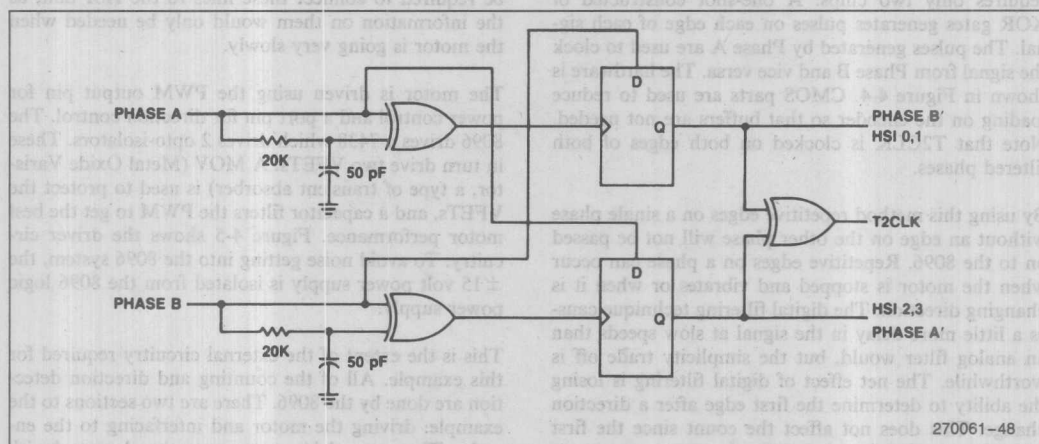


Figure 4-4. Schematic of Optical Encoder to 8096 Interface

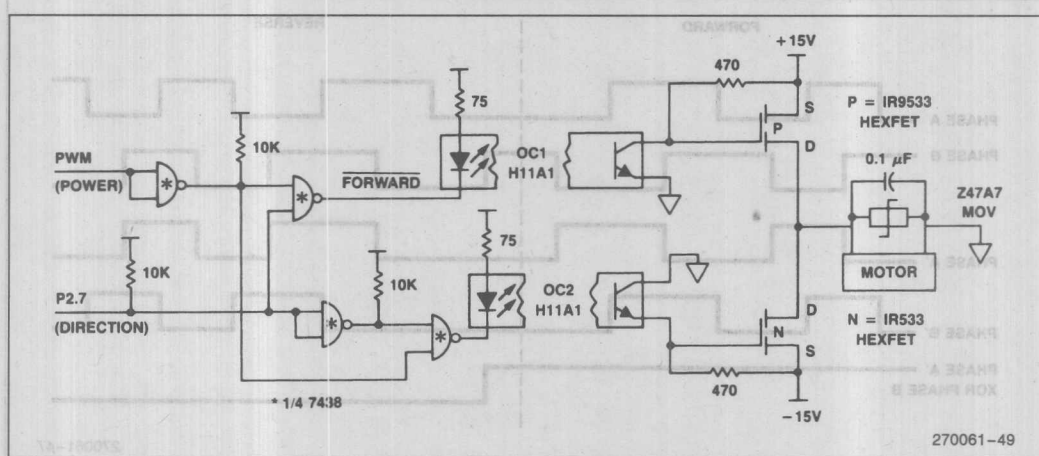


Figure 4-5. Motor Driver Circuitry

motor would decelerate smoothly until the time between encoder edges was around 100 microseconds. At this point the motor would either continue to decelerate slowly, or would suddenly stop and reverse. The latter case is the one that was most problematic.

After a brief overview, each section of the program will be described separately, with the complete listing included in the Appendix D. In order to make debugging easier, as well as to provide insight into how the program is working, I/O port 1 is used to indicate the program status. This information consists of which routine the program is in and under which mode it is operating. The main program sections are: Main loop, HSI interrupt, Timer 2 check, and Motor drive. There are also minor sections such as initialization, timer overflow handling, and software timer handling. Tying everything together is some overhead and glue. Where the glue is not obvious it will be discussed, otherwise it can be derived from the listings.

The program is a main loop which does nothing except serve as a place for the program to go when none of the interrupt routines are being run. All of the processing is done on an interrupt basis.

There are three basic software modes which are invoked depending on the speed of the motor. The modes referred to as 0, 1 and 2, in order from slowest to fastest operation. When the program is running the operating

mode is indicated by the lower 2 bits of Port 1, with the following coding:

P1.0	P1.1	Mode	Description
0	0	0	HSI looks at every edge
1	0	1	HSI looks at Phase A edges only
0	1	2	Timer 2 used instead of HSI
1	1	2	(alternate form of above)

The example is easiest to see if mode 2 is described first, followed by mode 1 then mode 0. In mode 2 Timer 2 is used to count edges on the incoming signal. A software timer routine, which is actually run using HSO.0, uses the Timer 2 value to update a LONG (32-bit) software counter labeled *POSITION*. The HSO routine runs every 260 microseconds. The HSO.0 interrupt is used instead of an actual software timer because of the ability to easily unmask it while other software timer routines are running.

In the code in Listing 4-7, the mode is first determined. For the first pass ignore the code starting with the label *in_mode_1*. Starting with *in_mode_2* the counter is incremented or decremented based on bit zero of *DIRECT*. If *DIRECT.0* = 0 the motor is going backward, if it is a 1 the motor is going forward. Next the count difference is checked to see if it is slow enough to go into mode 1. If not the routine returns to the code it was running when the interrupt occurred.

```

;=====
; SOFTWARE TIMER ROUTINE 0
;=====
; NOW USING HSO.0 TO TRIGGER
;=====
CSEG AT 2200H

hso_exec_int:    ; Check mode -- Update position in mode 2

    PUSHF
    ldb         HSO_COMMAND, #30H
    add         HSO_TIME, TIMER1, HSO0_dly

    orb         port1, #00100000B
    ld          timer_2, TIMER2
    jbs         port1, 1, in_mode2

in_mode1:
    sub         tmp1, timer_2, old_t2
    cmp         tmp1, #2
    jh          end_swt0
    set_mode0:
    jbc         port1, 0, end_swt0
    andb        port1, #11111100B
    ldb         ioc0, #01010101B
    ldb         last_stat, zero
    br          end_swt0

;=====
; Check count difference in tmp1
; if already in mode 0
; Clear P1.0, P1.1 (set mode 0)
; enable all HSI
;=====
    set         P1.5
    ; if already in mode 0
    ; Clear P1.0, P1.1 (set mode 0)
    ; enable all HSI
    ;=====
    270061-50

```

Listing 4-7. Motor Control HSO.0 Timer Routine


```

in_mode2:      sub    delta_p,timer_2,tmr2_old      ; get timer2 count difference
               ld      tmr2_old,timer_2

               jbc     direct,0,in_rev

in_fwd:        add     position,delta_p
               addc    position+2,zero
               brc     chk_mode

in_rev:        sub     position,delta_p
               subc    position+2,zero

chk_mode:      sub     tmpl,Timer_2,old_t2          ; Check count difference in tmpl
               cmp     tmpl,$55                     ; set model if count is too low
               jgt     end_sw0                       ; count <= 5

set_model:     andb    Port1,$11111018             ; Clear Pl.1, set Pl.0 (set mode 1)
               orb     Port1,$000000018
               ldb     IO0,$000001018               ; enable HSI 0 and 1
               ld      zero,HSI_TIME
               sub     last1-time,Timer1,min_hall   ; set up 0 to (Time-last2_time)>min_hsil on next HSI
               clr_hsi:
               ld      ZERO,HSI_TIME
               andb    io1_bak,$0101111118         ; clear bit 7
               orb     io1_bak,io1
               jbs     io1_bak,7,clr_hsi             ; If hsi is triggered then clear hsi

end_sw0:       ld      old_t2,TIMER_2
               andb    port1,$11011118
               POPF
               ret

```

Listing 4-7. Motor Control HSO.0 Timer Routine (Continued)

If the pulse rate is slow enough to go to mode 1, the transition is made by enabling HSI.0 and HSI.1. Both of these lines are connected to the same encoder line, with HSI.0 looking for rising edges and HSI.1 looking for falling edges. The *HSI_TIME* register is read to speed up clearing the HSI FIFO and the *LAST1_TIME* value is set up so the mode 1 routine does not immediately put the program into another mode. The HSI FIFO is then cleared, the Timer 2 value used throughout this routine is saved, and the routine returns.

This routine still runs in modes 0 and 1, but in an abbreviated form. The section of code starting with the label *in_mode1* checks to see if the pulses are coming in so slowly that both HSI lines can be checked. If this is the case then all of the HSIs are enabled and the program returns. This routine is the secondary method for going from mode 1 to mode 0, the primary method is by checking the time between edges during the HSI routine, which will be described later.

The HSO routine will enable mode 0 from mode 1 if two edges are not received every 260 microseconds. The primary method, (under the HSI routine), can only

enable mode 0 after an edge is received. This could cause a problem if the last 2 edges on Phase A before the encoder stops were too close to enable mode 0. If this happened, mode 0 would not be enabled until after the encoder started again, resulting in missed edges on Phase B. Using the HSO routine to switch from mode 1 to mode 0 eliminates this problem.

Figure 4-6 shows a state diagram of how the mode switching is done. As can be seen, there are two sources for most of the mode decisions. This helps avoid problems such as the one mentioned above.

When either Mode 1 or Mode 0 is enabled the HSI interrupt routine performs the counting of edges, while the HSO routine only ensures that the correct mode is running. The routines for modes 0 and 1 share the same initialization and completion sections, with the main body of code being different.

The initialization routine is similar to many HSI routines. The flags are checked to ensure that the HSI FIFO data is valid, and then the FIFO is read. Next, the main body of code (for either mode 0 or mode 1) is

run. At the end time and count values are saved and the holding register is checked for another event. Listing 4-8 contains the initialization and completion sections of the HSI routine.

Listing 4-9 is the main body of the Mode 1 routine. Before any calculations are done in Mode 1, the incoming pulse period is measured to see if it is too fast or too slow for mode 1. The time period between two edges is used so that the duty cycle of the waveform will not affect mode switching. If it is determined that Mode 2 should be set, Port 1.1 is set, all of the HSI lines are disabled, and the HSI fifo is cleared. If Mode 0 is to be set all of the HSI lines are enabled and the variable *LAST_STAT* is cleared. *LAST_STAT* = 0 is used as a flag to indicate the first HSI interrupt in Mode 0 after Mode 1. After the mode checking and setting are complete the incremental value in Timer 2 is used to update

POSITION. The program then returns to the completion section of the routine.

There is a lot more code used in Mode 0 than in Mode 1, most of which is due to the multiple jump statements that determine the current and previous state of the HSI pins. In order to save execution time several blocks of code are repeated as can be seen in Listing 4-10. The first determination is that of which edge had occurred. If a Phase A edge was detected the *LAST1_TIME* and *LAST2_TIME* variables are updated so a reference to the pulse frequency will be available. These are the same variables used under Mode 1. A test is also made to see if the edges are coming fast enough to warrant being in Mode 1, if they are, the switch is made. If the last edge detected was on Phase B, the information is used only to determine direction.

```

In_mode_1:                ; mode 1 HSI routine

    andb    tmp1,hsi_s0,$0101010000B
    jne     no_cnt

cmp_time:                  ; Procedure which sets mode 1 also
                           ; sets times to pass the tests

    ld      last2_time,last1_time
    ld      last1_time,time

cmpl:    sub    tmp1,time,last2_time
    cmp     tmp1,min_hsil
    jh      check_max_time

set_mode_2:
    orb     Port1,$000000010B    ; Set Pl.1 (in mode 2)
    ldb     IOC0,$000000000B    ; Disable all HSI
    mt_hsi: ld      zero,hsi_time ; empty the hsi fifo
    andb    loasl_bak,$01111111B ; clear bit 7
    orb     loasl_bak,loasl
    jbs     loasl_bak,7,mt_hsi   ; If hsi is triggered then clear hsi
    br      done_chk

check_max_time:
    sub     tmp1,time,last2_time
    cmp     tmp1,max_hsil        ; max_hsil = addition to min_hsil for
    jnh     done_chk             ; total time

set_mode_0:
    andb    Port1,$11111100B    ; clear Pl.0,1 (set mode 0)
    ldb     IOC0,$01010101B    ; Enable all HSI
    ldb     last_stat,zero

done_chk:
    sub     delta_p,timer_2,tmr2_old ; get timer2 count difference
    jbc     direct,0,add_rev

add_fwd:
    add     position,delta_p
    addc    position+2,zero
    br      load_last

add_rev:
    sub     position,delta_p
    subc    position+2,zero
    br      load_last

$seject

```

270061-54

Listing 4-9. Motor Control Mode 1 Routines

<pre> in_mode_0: jbs hsl_s0,0,a_rise jbs hsl_s0,2,a_fall jbs hsl_s0,4,b_rise jbs hsl_s0,6,b_fall br no_cnt a_rise: ld last2_time,last1_time ld last1_time,time sub time,last2_time cmp time,min_hsl jh tst_statf ;set model orb Port1,\$000000001B ldb IOC0,\$000000101B tst statf jbs last_stat,6,going_fwd jbs last_stat,4,going_rev jbs last_stat,2,change_dir cmpb last_stat,zero je first_time br inp_err a_fall: ld last2_time,last1_time ld last1_time,time sub time,last2_time cmp time,min_hsl jh tst_statf ;set model orb Port1,\$000000001B ldb IOC0,\$000000101B tst statf jbs last_stat,4,going_fwd jbs last_stat,6,going_rev jbs last_stat,0,change_dir cmpb last_stat,zero je first_time br inp_err b_rise: jbs last_stat,0,going_fwd jbs last_stat,2,going_rev jbs last_stat,6,change_dir cmpb last_stat,zero je first_time br inp_err b_fall: jbs last_stat,2,going_fwd jbs last_stat,0,going_rev jbs last_stat,4,change_dir cmpb last_stat,zero je first_time br inp_err first_time: stb hsl_s0,last_stat br done_chk ; add delta position inp_err: br no_int change_dir: notb direct no_inc: jbc direct,0,going_rev going_fwd: orb PORT2,\$01000000B ldb direct,\$01 add position,\$01 addc position+2,zero br st_stat going_rev: andb PORT2,\$10111111B ldb direct,\$00 sub position,\$01 subc position+2,zero st_stat: stb hsl_s0,last_stat </pre>	<pre> ; Set P1.0 (in mode 1) ; Enable HSI 0 and 1 ; Set P2.6 ; direction = forward ; clear P2.6 ; direction = reverse </pre>
--	--

Listing 4-10. Motor Control Mode 0 Routines

270061-55

After mode correctness is confirmed and the *LASTx_TIME* values are updated the *LAST_STAT* (Last Status) variable is used to determine the current direction of travel. The *POSITION* value is then updated in the direction specified by the last two edges and the status is stored. Note that the first time in Mode 0 after being in Mode 1, the Mode 1 *done_chk* routine is used to update *POSITION*, instead of the routines *going_fwd* and *going_rev* from the Mode 0 section of code. The completion section of code is then executed.

Providing the PWM value to drive the motor is done by a routine running under Software Timer 1. The first section of code, shown in Listing 4-11a, has to do with calculating the position and timer errors. Listing 4-11b shows the next section of code where the power to be supplied to the motor is calculated. First the direction is checked and if the direction is reverse the absolute value of the error is taken. If the error is greater than 64K counts, the PWM routine is loaded with the maximum value. The next check is made to see if the motor

is close enough to the desired location that the power to it should be reversed, (i.e., enter the Braking mode). If the motor is very close to the position or has slowed to the point that is likely to turn around, the *Hold_Position mode* is entered.

The determination of which modes are selected under what conditions was done empirically. All of the parameters used to determine the mode are kept in RAM so they can be easily changed on the fly instead of by re-assembling the program. The parameters in the listing have been selected to make the motor run, but have not been optimized for speed or stability. A diagram of the modes is shown in Figure 4-7.

In the *Hold_Position* mode power is eased onto the motor to lock it into position. Since the motor could be stopped in this mode, some integral control is needed, as proportional control alone does not work well when the error is small and the load is large. The *BOOST* variable provides this integral control by increasing the output a fixed amount every time period in which the

Listing 4-11. Motor Control Software Timer 1 Routine

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!                                SOFTWARE TIMER ROUTINE 1                                !!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

CSEG AT 2600H

swt1_expired:

    pushf
    orb     port1,%10000000B           ; set port1.7
    ldb     int_mask,%00001101B        ; enable HST, Tcvf, HSO
    ldb     HSO_COMMAND,%39H
    add     HSO_TIME,TIMER1,swt1_dly

    ld      time_err+2,des_time+2      ; Calculate time & position error
    ld      pos_err+2,des_pos+2
    sub     time_err,des_time,time     ; values are set
    subc    time_err+2,time+2
    sub     pos_err,des_pos,position
    subc    pos_err+2,position+2

    EI

    sub     time_delta,last_time_err,time_err
    ld      last_time_err,time_err

    sub     pos_delta,last_pos_err,pos_err
    ld      last_pos_err,pos_err

!!!!!!                                Time_err = Desired time to finish - current time
!!!!!!                                Pos_err   = Desired position to finish - current position
!!!!!!                                Pos_delta = Last position error - Current position error
!!!!!!                                Time_delta = Last time error - Current time error
!!!!!!                                note that errors should get smaller so deltas will be
!!!!!!                                positive for forward motion (time is always forward)

```

270061-56

Listing 4-11a. Motor Control Software Position Counter

```

chk_dir: cmp     pos_err+2,zero
        jge     go_forward

go_backward: neg     pos_err      ; Pos_err = ABS VAL (pos_err)
          ldb     pwm_dir,#00h
          cmp     pos_err+2,#0ffffh
          jne     ld_max
          br      chk_brk

go_forward: ldb     pwm_dir,#01h
          cmp     pos_err+2,zero
          je      chk_brk

ld_max: ldb     pwm_pwr,max_pwr
        br      chk_sanity

chk_brk: ; Position_Error now = ABS(pos_err)
        cmp     pos_err,pos_pnt
        jnh     hold_position ; position_error < position_control_point
        cmp     pos_err,brk_pnt
        jh      ld_max ; position_error > brake_point

braking: cmp     pos_delta,zero
        jge     chk_delta
        neg     pos_delta

chk_delta: cmp     pos_delta,vel_pnt ; velocity = pos_delta/sample_time
          jnh     hold_position ; jmp if ABS(velocity) < vel_pnt

brake: ldb     pwm_pwr,max_brk
       ldb     tmp,direct ; If braking apply power in opposite
       notb    tmp ; direction of current motion
       ldb     pwm_dir,tmp

       br      ld_pwr

Hold_position: ; position hold mode
        cmp     pos_err,#02
        jh      calc_out ; if position error < 2, then turn off power
        clr     tmp+2
        clr     boost
        BR      output

calc_out: mulub    tmp,max_hold,#255
          mulu     tmp,pos_err ; Tmp = pos_err * max_hold
          cmp     pos_delta,zero
          jne     no_bst
          boost,#04 ; Boost is integral control
          add     tmp+2,boost ; TMP+2 = MSB(pos_err*max_hold)
          br      ck_max
          boost    ck_max
          boost    tmp+2,max_hold
          output   tmp+2,max_hold
          maxed: ld  tmp+2,max_hold
          output: ldb pwm_pwr,tmp+2

chk_sanity: br      ld_pwr

ld_pwr: ldb     rpwr,pwm_pwr
        notb    rpwr
        jbs     pwm_dir,0,p2fwd

p2bkwd: DI
        andb    port2,#01111111b
        ldb     pwm_control,rpwr ; clear P2.7

p2fwd: DI
        br      pwrset
        DI
        orb     port2,#10000000b ; set P2.7
        ldb     pwm_control,rpwr
        EI

```

Listing 4-11b: Motor Control Power Algorithm

270061-57

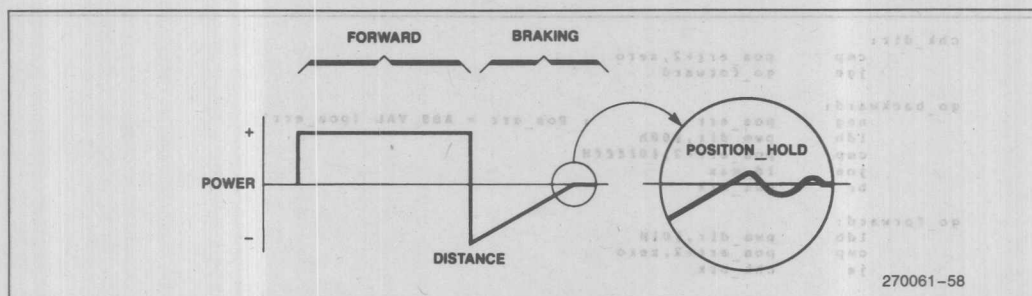


Figure 4-7. Motor Control Modes

error does not get smaller. Once the error does get smaller, usually because the motor starts moving, BOOST is cleared.

A sanity check can be performed at this point to double check that the 8096 has proper control of the motor. In the example the worst that can happen is the proto-

```

pwrset:  cmp     time_err+2,zero    ; do pos_table when err is negative
        jgt     end_p
        br      end_p
        cmp     nxt_pos,(32+pos_table)
        jlt     get_vals          ; jump if lower
        ld      nxt_pos,pos_table
        clr     time+2
get_vals:
        ld      des_pos,[nxt_pos]+
        ld      des_pos+2,[nxt_pos]+
        ld      des_time+2,[nxt_pos]+
        ld      max_pwr,[nxt_pos]+
        add     des_pos,offset
        addc    des_pos+2,zero
        sub     last_pos_err,des_pos,position

end_p:   andb    port1,#01111111B    ; clear P1.7
        popf
        ret

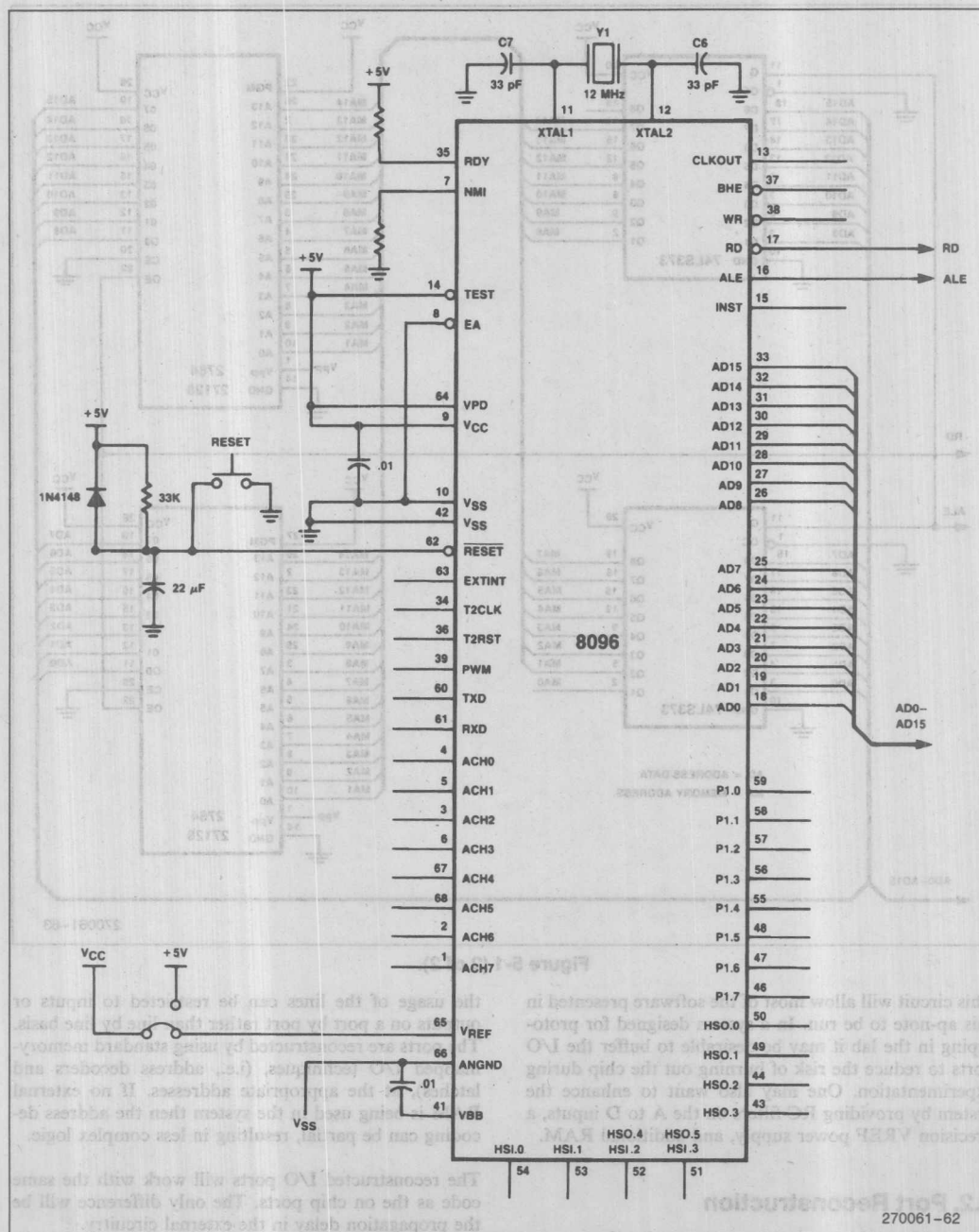
pos_table:
        dcl     00000000H           ; position 0
        dcw     0020H, 0080H        ; next time, power
        dcl     0000c000H           ; position 1
        dcw     0040H, 0040H        ; next time, power
        dcl     00000000H           ; position 2
        dcw     0060H, 00c0H        ; next time, power
        dcl     0FFFF800H           ; position 3
        dcw     0080H, 0080H        ; next time, power

        dcl     00000800H           ; position 4
        dcw     0058H, 0080H        ; next time, power
        dcl     00003000H           ; position 5
        dcw     0070H, 00ffH        ; next time, power
        dcl     00000000H           ; position 6
        dcw     0090H, 00f0H        ; next time, power
        dcl     00000000H           ; position 7
        dcw     0091H, 00f0H        ; next time, power

```

270061-59

Listing 4-12. Motor Control Next Position Lookup



one contains the odd bytes, and the addressing is not fully decoded. This means that the addressing on a 2764 will be such that the lower 4K of each EPROM is mapped at 0000H and 4000H while the upper

4K is mapped at 2000H. If the program being loaded is 16 Kbytes long the first half is loaded into the second half of the 2764s and vice versa. A similar situation exists when using 27128s.

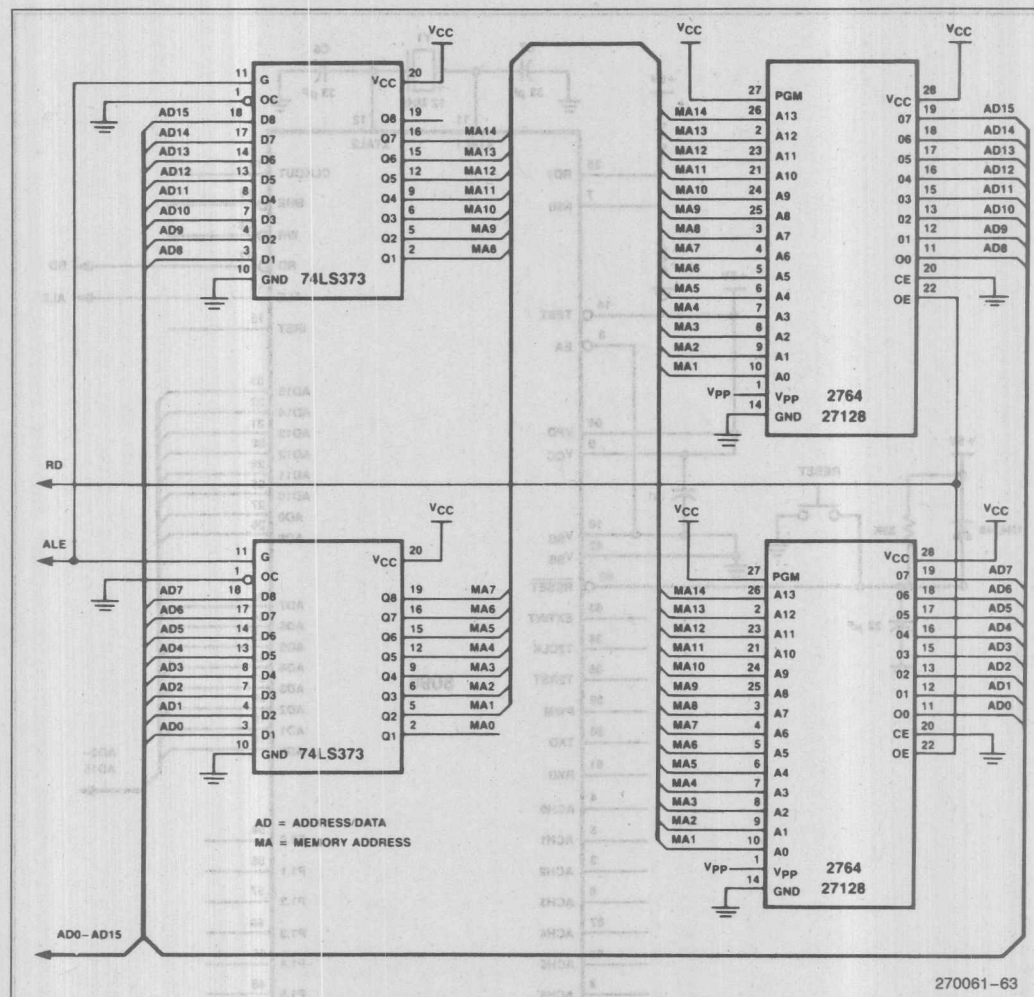


Figure 5-1 (2 of 2).

This circuit will allow most of the software presented in this ap-note to be run. In a system designed for prototyping in the lab it may be desirable to buffer the I/O ports to reduce the risk of burning out the chip during experimentation. One may also want to enhance the system by providing RC filters on the A to D inputs, a precision VREF power supply, and additional RAM.

5.2. Port Reconstruction

If it is desired to fully emulate a 8396 then I/O ports 3 and 4 must be reconstructed. It is easiest to do this if

the usage of the lines can be restricted to inputs or outputs on a port by port rather than line by line basis. The ports are reconstructed by using standard memory-mapped I/O techniques, (i.e., address decoders and latches), at the appropriate addresses. If no external RAM is being used in the system then the address decoding can be partial, resulting in less complex logic.

The reconstructed I/O ports will work with the same code as the on chip ports. The only difference will be the propagation delay in the external circuitry.

6.0 CONCLUSION

An overview of the MCS-96 family has been presented along with several simple examples and a few more complex ones. The source code for all of these programs are available in the Insite Users Library using order code AE-16. Additional information on the 8096 can be found in the Microcontroller Handbook and it is recommended that this book be in your possession before attempting any work with the MCS-96 family of products. Your local Intel sales office can assist you in getting more information on the 8096 and its hardware and software development tools.

7.0 BIBLIOGRAPHY

1. MSC-96 Macro Assembler User's Guide, Intel Corporation, 1983.
2. Microcontroller Handbook (1985), Intel Corporation, 1984.
3. MSC-96 Utilities User's Guide, Intel Corporation, 1983.
4. PL/M-96 User's Guide, Intel Corporation, 1983.

Line	Address	Op Code	Op Name	Comment
01	5000	0000	LD R0, #0	
02	5001	0000	LD R0, #0	
03	5002	0000	LD R0, #0	
04	5003	0000	LD R0, #0	
05	5004	0000	LD R0, #0	
06	5005	0000	LD R0, #0	
07	5006	0000	LD R0, #0	
08	5007	0000	LD R0, #0	
09	5008	0000	LD R0, #0	
10	5009	0000	LD R0, #0	
11	500A	0000	LD R0, #0	
12	500B	0000	LD R0, #0	
13	500C	0000	LD R0, #0	
14	500D	0000	LD R0, #0	
15	500E	0000	LD R0, #0	
16	500F	0000	LD R0, #0	
17	5010	0000	LD R0, #0	
18	5011	0000	LD R0, #0	
19	5012	0000	LD R0, #0	
20	5013	0000	LD R0, #0	
21	5014	0000	LD R0, #0	
22	5015	0000	LD R0, #0	
23	5016	0000	LD R0, #0	
24	5017	0000	LD R0, #0	
25	5018	0000	LD R0, #0	
26	5019	0000	LD R0, #0	
27	501A	0000	LD R0, #0	
28	501B	0000	LD R0, #0	
29	501C	0000	LD R0, #0	
30	501D	0000	LD R0, #0	
31	501E	0000	LD R0, #0	
32	501F	0000	LD R0, #0	
33	5020	0000	LD R0, #0	
34	5021	0000	LD R0, #0	
35	5022	0000	LD R0, #0	
36	5023	0000	LD R0, #0	
37	5024	0000	LD R0, #0	
38	5025	0000	LD R0, #0	
39	5026	0000	LD R0, #0	
40	5027	0000	LD R0, #0	
41	5028	0000	LD R0, #0	
42	5029	0000	LD R0, #0	
43	502A	0000	LD R0, #0	
44	502B	0000	LD R0, #0	
45	502C	0000	LD R0, #0	
46	502D	0000	LD R0, #0	
47	502E	0000	LD R0, #0	
48	502F	0000	LD R0, #0	
49	5030	0000	LD R0, #0	
50	5031	0000	LD R0, #0	
51	5032	0000	LD R0, #0	
52	5033	0000	LD R0, #0	
53	5034	0000	LD R0, #0	
54	5035	0000	LD R0, #0	
55	5036	0000	LD R0, #0	
56	5037	0000	LD R0, #0	
57	5038	0000	LD R0, #0	
58	5039	0000	LD R0, #0	
59	503A	0000	LD R0, #0	
60	503B	0000	LD R0, #0	
61	503C	0000	LD R0, #0	
62	503D	0000	LD R0, #0	
63	503E	0000	LD R0, #0	
64	503F	0000	LD R0, #0	
65	5040	0000	LD R0, #0	
66	5041	0000	LD R0, #0	
67	5042	0000	LD R0, #0	
68	5043	0000	LD R0, #0	
69	5044	0000	LD R0, #0	
70	5045	0000	LD R0, #0	
71	5046	0000	LD R0, #0	
72	5047	0000	LD R0, #0	
73	5048	0000	LD R0, #0	
74	5049	0000	LD R0, #0	
75	504A	0000	LD R0, #0	
76	504B	0000	LD R0, #0	
77	504C	0000	LD R0, #0	
78	504D	0000	LD R0, #0	
79	504E	0000	LD R0, #0	
80	504F	0000	LD R0, #0	
81	5050	0000	LD R0, #0	
82	5051	0000	LD R0, #0	
83	5052	0000	LD R0, #0	
84	5053	0000	LD R0, #0	
85	5054	0000	LD R0, #0	
86	5055	0000	LD R0, #0	
87	5056	0000	LD R0, #0	
88	5057	0000	LD R0, #0	
89	5058	0000	LD R0, #0	
90	5059	0000	LD R0, #0	
91	505A	0000	LD R0, #0	
92	505B	0000	LD R0, #0	
93	505C	0000	LD R0, #0	
94	505D	0000	LD R0, #0	
95	505E	0000	LD R0, #0	
96	505F	0000	LD R0, #0	
97	5060	0000	LD R0, #0	
98	5061	0000	LD R0, #0	
99	5062	0000	LD R0, #0	
100	5063	0000	LD R0, #0	

APPENDIX A BASIC SOFTWARE EXAMPLES

SERIES-III MCS-96 MACRO ASSEMBLER, V1 0

SOURCE FILE: F3:INTER1 A96

OBJECT FILE: F3:INTER1 OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: NOSB

ERR LOC	OBJECT	LINE	SOURCE STATEMENT
		1	*TITLE('INTER1 A96: Interpolation routine 1')
		2	***** 8096 Assembly code for table lookup and interpolation
		3	
		4	*INCLUDE('F0: DEMO96. INC') ; Include demo definitions
=1		5	*nolist ; Turn listing off for include file
=1		53	; End of include file
	0022	54	
		55	RSEG at 22H
		56	
	0022	57	IN_VAL: dsb 1 ; Actual Input Value
	0024	58	TABLE_LOW: dsb 1
	0026	59	TABLE_HIGH: dsb 1
	0028	60	IN_DIF: dsb 1 ; Upper Input - Lower Input
	0028	61	IN_DIFB equ IN_DIF : byte
	002A	62	TAB_DIF: dsb 1 ; Upper Output - Lower Output
	002C	63	OUT: dsb 1
	002E	64	RESULT: dsb 1
	0030	65	OUT_DIF: dsb 1 ; Delta Out
		66	
		67	
	2080	68	CSEG at 2080H
		69	
	2080 A100011B	70	LD SP, #100H
		71	
	2084 B0221C	72	look: LDB AL, IN_VAL ; Load temp with Actual Value
	2087 18031C	73	SHRB AL, #3 ; Divide the byte by 8
	208A 71FE1C	74	ANDB AL, #11111110B ; Insure AL is a word address
		75	; This effectively divides AL by 2
		76	; so AL = IN_VAL/16
		77	
	208D AC1C1C	78	LDBZE AX, AL ; Load byte AL to word AX
	2090 A31D002124	79	LD TABLE_LOW, TABLE [AX] ; TABLE_LOW is loaded with the value
		80	; in the table at table location AX
		81	

270061-64

A.1. Table Lookup 1

19-102

A.1. Table Lookup 1 (Continued)

19-103

```

2095 A31D022126      82      LD      TABLE_HIGH, (TABLE+2)[AX] ; TABLE_HIGH is loaded with the
                        83                          ; value in the table at table
                        84                          ; location AX+2
                        85                          ; (The next value in the table)
                        86
209A 4824262A      87      SUB      TAB_DIF, TABLE_HIGH, TABLE_LOW
                        88                          ; TAB_DIF=TABLE_HIGH-TABLE_LOW
                        89
209E 510F222B      90      ANDB     IN_DIFB, IN_VAL, #0FH ; IN_DIFB=least significant 4 bits
                        91                          ; of IN_VAL
20A2 AC282B      92      LDBZ     IN_DIF, IN_DIFB ; Load byte IN_DIFB to word IN_DIF
                        93
20A5 FE4C2A2B30      94      MUL      OUT_DIF, IN_DIF, TAB_DIF ; Output_difference =
                        95                          ; Input_difference*Table_difference
                        96
20AA 0E0430      97      SHRAL    OUT_DIF, #4 ; Divide by 16 (2**4)
                        98
20AD 4424302C      99      ADD      OUT, OUT_DIF, TABLE_LOW ; Add output difference to output
                        100                          ; generated with truncated IN_VAL
                        101                          ; as input
20B1 0A042C      102      SHRA     OUT, #4 ; Round to 12-bit answer
20B4 A4002C      103      ADDC     OUT, zero ; Round up if Carry = 1
                        104
20B7 C02E2C      105      no_inc: ST      OUT, RESULT ; Store OUT to RESULT
                        106
20BA 27C8      107      BR      look ; Branch to "look:"
                        108
2100      109      cseg      AT 2100H
                        110
2100 000000200034004C      111      table: DCW     0000H, 2000H, 3400H, 4C00H ; A random function
2108 005D006A0072007B      112      DCW     5D00H, 6A00H, 7200H, 7B00H
2110 007B007D0076006D      113      DCW     7B00H, 7D00H, 7600H, 6D00H
2118 005D004B00340022      114      DCW     5D00H, 4B00H, 3400H, 2200H
2120 0010      115      DCW     1000H
                        116
2122      117      END
                        118
ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

```

270061-65
 EMM FOR OBJECT
 COMINCS SPECIFIED IN INADCVION COMMAND WORD
 OBJECT LIFE 13 INIENS DBI
 SOURCE LIFE 13 INIENS VAP
 SERIES III MCB-89 MACHO VERSION 1.0

SERIES-III MCS-96 MACRO ASSEMBLER, V1.0

SOURCE FILE F3.INTER2.A96

OBJECT FILE F3.INTER2.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND NOSB

```

ERR LOC OBJECT LINE SOURCE STATEMENT
1 $TITLE('INTER2.A96 Interpolation routine 2')
2
3 ..... B096 Assembly code for table lookup and interpolation
4 ..... Using tabled values in place of division
5
6 $INCLUDE('FO.DEMO96.INC'), Include demo definitions
=1 7 $nolist ; Turn listing off for include file
=1 55 END ; End of include file
56
0024 0010 57 RSEG at 24H 1000H
58
59 IN_VAL: 800H dsb 1 ; Actual Input Value
60 TABLE_LOW: 800H dsb 1 ; Table value for function
61 TABLE_INC: 800H dsb 1 ; Incremental change in function
62 IN_DIF: 800H dsb 1 ; Upper Input - Lower Input
63 IN_DIF: equ IN_DIF : byte
64 OUT: dsb 1
65 RESULT: dsb 1
66 OUT_DIF: dsb 1 ; Delta Out
67
68 CSEG at 2080H
69
70 LD SP, #100H ; Initialize SP to top of reg. file
71
72 look: LDB AL, IN_VAL ; Load temp with Actual Value
73 SHRB AL, #3 ; Divide the byte by 8
74 ANDB AL, #11111110B ; Insure AL is a word address
75 ; This effectively divides AL by 2
76 ; so AL = IN_VAL/16
77 LDBZ AX, AL ; Load byte AL to word AX
78
79 LD TABLE_LOW, VAL_TABLE[AX] ; TABLE_LOW is loaded with the value
80 ; in the value table at location AX
81
82 LD TABLE_INC, INC_TABLE[AX] ; TABLE_INC is loaded with the value
83 ; in the increment table at
84 ; location AX*2 HIGH-16BET.COM
85
86
87
88
89
90
91
92
93

```

270061-66

```

209A 510F242A      87      ANDB      IN_DIFB, IN_VAL, #0FH      ; IN_DIFB=least significant 4 bits
                                     88                      ; of IN_VAL
209E AC2A2A        89      LDBZ     IN_DIF, IN_DIFB      ; Load byte IN_DIFB to word IN_DIF
                                     90
20A1 FE4C2B2A30    91      MUL      OUT_DIF, IN_DIF, TABLE_INC
                                     92                      ; Output_difference =
                                     93                      ; Input_difference*Incremental_change
20A6 4426302C      94
                                     95      ADD      OUT, OUT_DIF, TABLE_LOW ; Add output difference to output
                                     96                      ; generated with truncated IN_VAL
                                     97                      ; as input
20AA 0B042C        98      SHR      OUT, #4      ; Round to 12-bit answer
20AD A4002C        99      ADDC     OUT, zero      ; Round up if Carry = 1
                                     100
20B0 C02E2C        101     no_inc: ST      OUT, RESULT      ; Store OUT to RESULT
20B3 27CF          102     BR      look      ; Branch to "look:"
                                     103
                                     104
2100              105     cseg      AT 2100H
                                     106
2100              107     val_table:
2100 000000200034004C 108     DCW      0000H, 2000H, 3400H, 4C00H ; A random function
2108 005D006A0072007B 109     DCW      5D00H, 6A00H, 7200H, 7B00H
2110 007B007D0076006D 110     DCW      7B00H, 7D00H, 7600H, 6D00H
2118 005D004B00340022 111     DCW      5D00H, 4B00H, 3400H, 2200H
2120 0010          112     DCW      1000H
2122              113     inc_table:
2122 0002400180011001 114     DCW      0200H, 0140H, 0180H, 0110H ; Table of incremental
212A D000800060003000 115     DCW      00D0H, 00B0H, 0060H, 0030H ; differences
2132 200090FF70FF00FF 116     DCW      00020H, 0FF90H, 0FF70H, 0FF00H
213A E0FE90FEE0FEE0FE 117     DCW      0FEE0H, 0FE90H, 0FEE0H, 0FEE0H
                                     118
2142              119     END
ASSEMBLY COMPLETED, NO ERROR(S) FOUND

```

270061-67

SERIES-III PL/M-96 V1 0 COMPILATION OF MODULE PLMEX
 OBJECT MODULE PLACED IN : F3 PLMEX1 OBJ
 COMPILER INVOKED BY: PLM96.B6 F3.PLMEX1.P96 CODE

\$TITLE('PLMEX1: PLM-96 Example Code for Table Lookup')

/* PLM-96 CODE FOR TABLE LOOK-UP AND INTERPOLATION */

```

1      PLMEX:      DO;

2      1      DECLARE IN_VAL      WORD      PUBLIC;
3      1      DECLARE TABLE_LOW  INTEGER   PUBLIC;
4      1      DECLARE TABLE_HIGH INTEGER   PUBLIC;
5      1      DECLARE TABLE_DIF  INTEGER   PUBLIC;
6      1      DECLARE OUT         INTEGER   PUBLIC;
7      1      DECLARE RESULT      INTEGER   PUBLIC;
8      1      DECLARE OUT_DIF     LONGINT   PUBLIC;
9      1      DECLARE TEMP        WORD      PUBLIC;

10     1      DECLARE TABLE(17)  INTEGER DATA ( /* A random function */
        0000H, 2000H, 3400H, 4C00H,
        5D00H, 6A00H, 7200H, 7B00H,
        7B00H, 7D00H, 7600H, 6D00H,
        5D00H, 4B00H, 3400H, 2200H,
        1000H);

11     1      DMPY: PROCEDURE (A,B) LONGINT EXTERNAL;
12     2      DECLARE (A,B) INTEGER;
13     2      END DMPY;

14     1      LOOP
        TEMP=SHR(IN_VAL,4); /* TEMP is the most significant 4 bits of IN_VAL */

15     1      TABLE_LOW=TABLE(TEMP); /* If "TEMP" was replaced by "SHR(IN_VAL,4)" */
16     1      TABLE_HIGH=TABLE(TEMP+1); /* The code would work but the 8096 would */
        /* do two shifts */

17     1      TABLE_DIF=TABLE_HIGH-TABLE_LOW;

18     1      OUT_DIF=DMPY(TABLE_DIF,SIGNED(IN_VAL AND 0FH)) /16;

19     1      OUT=SAR((TABLE_LOW+OUT_DIF),4); /* SAR performs an arithmetic right shift,
        in this case 4 places are shifted */
  
```

270061-68

```

20 1      IF CARRY=0 THEN RESULT=OUT; /* Using the hardware flags must be done */
22 1      ELSE RESULT=OUT+1;          /* with care to ensure the flag is tested */
                                         /* in the desired instruction sequence */
23 1      GOTO LOOP;

      /* END OF PLM-96 CODE */

24 1      END;

```

270061-69

PL/M-96 COMPILER PLMEX1: PLM-96 Example Code for Table Lookup
ASSEMBLY LISTING OF OBJECT CODE

```

PLM-96 COMPILER PLMEX1: PLM-96 Example Code for Table Lookup
ASSEMBLY LISTING OF OBJECT CODE

STATEMENT 14
0022      PLMEX: LD SP, #STACK
0022 A1000018 R
0026      LOOP: LD TEMP, IN_VAL
0026 A00010 R
0029 0B0410 R SHR TEMP, #4H
; STATEMENT 15
002C 4410101C R ADD TMP0, TEMP, TEMP
0030 A31D000002 R LD TABLE_LOW, TABLE[TMP0]
; STATEMENT 16
0035 A31D020004 R LD TABLE_HIGH, TABLE+2H[TMP0]
; STATEMENT 17
003A 48020406 R SUB TABLE_DIF, TABLE_HIGH, TABLE_LOW
; STATEMENT 18
003E C806 R PUSH TABLE_DIF
0040 410F00001C R AND TMP0, IN_VAL, #0FH
0045 C81C R PUSH TMP0
0047 EF0000 E CALL DMPY
004A 0E041C R SHRAL TMP0, #4H
004D A01E0E R LD OUT_DIF+2H, TMP2
0050 A01C0C R LD OUT_DIF, TMP0
; STATEMENT 19
0053 A00220 R LD TMP4, TABLE_LOW
0056 0620 R EXT TMP4
0058 641C20 R ADD TMP4, TMP0
005B A41E22 R ADDC TMP4, TMP2
005E 0E0420 R SHRAL TMP4, #4H
0061 A02008 R LD OUT, TMP4
; STATEMENT 20
0064 B1FF1C R LDB TMP0, #0FFH
0067 DB02 R BC #0003
0069 111C R CLRB TMP0
006B @0003: @0003:
; STATEMENT 21
007E D103 R SNE #0001
007B 0B1C00 R CMB #0001

```

270061-70

```

006B 981C00      CMPB  RO, TMP0
006E D705        BNE   @0001

0070 A0200A      R  @0003: ; STATEMENT 21
0073 2005        LD    RESULT, TMP4
0075 B1447C      BR    @0002 ; STATEMENT 22
0077 A0080A      R  @0001: ; STATEMENT 23
0078 070A        R  LD    RESULT, OUT
007A 27AA        R  INC   RESULT
007C 27AA        @0002: ; STATEMENT 24
007E 27AA        BR    LOOP
0080 27AA        END

MODULE INFORMATION:
CODE AREA SIZE      = 005AH 90D
CONSTANT AREA SIZE  = 0022H 34D
DATA AREA SIZE      = 0000H 0D
STATIC REGS AREA SIZE = 0012H 18D

PL/M-96 COMPILER PLMEX1 PLM-96 Example Code for Table Lookup
ASSEMBLY LISTING OF OBJECT CODE

OVERLAYABLE REGS AREA SIZE = 0000H 0D
MAXIMUM STACK SIZE        = 0006H 6D
48 LINES READ

PL/M-96 COMPILATION COMPLETE 0 WARNINGS, 0 ERRORS

```

270061-71

MCS-96 MACRO ASSEMBLER MULT.APT: 16*16 multiply procedure for PLM-96

SERIES-III MCS-96 MACRO ASSEMBLER, V1.0

SOURCE FILE: :F3:MULT.A96

OBJECT FILE: :F3:MULT.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: NOSB

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
			1	\$TITLE('MULT.APT: 16*16 multiply procedure for PLM-96')
			2	
			3	
	0018		4	SP EQU 18H:word
			5	
	0000		6	rseg
			7	EXTRN PLMREG :long
			8	
	0000		9	cseg
			10	
			11	PUBLIC DMPY ; Multiply two integers and return a
			12	; longint result in AX, DX registers
***	0000	CC04	13	DMPY: POP PLMREG+4 ; Load return address
***	0002	CC00	14	POP PLMREG ; Load one operand
***	0004	FE6E1900	15	MUL PLMREG,[SP]+ ; Load second operand and increment SP
***	0008	E3041VCH	16	BR [PLMREG+4] ; Return to PLM code.
***	000A	REG	17	END
ASSEMBLY COMPLETED,				NO ERROR(S) FOUND.

270061-72

RECHEN1 WAP FOR E3 670001 087(67HEX):

670001 F18(67HEX) 1103182
 E3 670001 087(67HEX) 15152182
 E3 670001 087(67HEX) 15152182
 INH01 W000122 INC0002

60W(5000H-000H)
 CONTROLS SPECIFIED IN INVOCATION COMMAND:
 OBJECT FILE: E3 670001 087
 INH01 LIFE: E3 670001 087 E3 670001 087 670001 F18

Copyright 1983 Intel Corporation
 SERIES-III MCS-96 MICROVISION AND FINNEX AS G

SERIES-III MCS-96 RELOCATOR AND LINKER, V2.0
Copyright 1983 Intel Corporation

INPUT FILES: :F3:PLMEX1.OBJ, :F3:MULT.OBJ, PLM96.LIB
OUTPUT FILE: :F3:PLMOUT.OBJ
CONTROLS SPECIFIED IN INVOCATION COMMAND:
ROM(2080H-3FFFFH)

INPUT MODULES INCLUDED:
:F3:PLMEX1.OBJ(PLMEX) 12/25/84
:F3:MULT.OBJ(MULT) 12/25/84
PLM96.LIB(PLMREG) 11/02/83

SEGMENT MAP FOR :F3:PLMOUT.OBJ(PLMEX):

TYPE	BASE	LENGTH	ALIGNMENT	MODULE NAME
**RESERVED*	0000H	001AH		
*** GAP ***	001AH	0002H		
REG	001CH	0008H	ABSOLUTE	PLMREG
REG	0024H	0012H	WORD	PLMEX
STACK	0036H	0006H	WORD	
*** GAP ***	003CH	2044H		
CODE	2080H	0003H	ABSOLUTE	PLMEX
*** GAP ***	2083H	0001H		
CODE	2084H	007CH	WORD	PLMEX
CODE	2100H	000AH	BYTE	MULT
*** GAP ***	210AH	DEF6H		

270061-73

ATTRIBUTES	VALUE	NAME
REG	WORD	0024H
REG	INTEGER	0026H
REG	INTEGER	0028H
REG	INTEGER	002AH
REG	INTEGER	002CH
REG	INTEGER	002EH
REG	LONGINT	0030H
REG	WORD	0034H
CODE	ENTRY	2100H
REG	LONG	001CH
NULL	NULL	003CH
NULL	NULL	1FC4H

RL96 COMPLETED, 0 WARNING(S), 0 ERROR(S)

270061-74

SERIES-III MCS-96 MACRO ASSEMBLER, V1 0

SOURCE FILE .F3:PULSE.A96

OBJECT FILE .F3:PULSE.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND NOSB

ERR LOC	OBJECT	LINE	SOURCE STATEMENT
		1	\$TITLE('PULSE.A96: Measuring pulses using the HSI unit')
		2	
		3	\$INCLUDE(DEMO96 INC)
		=1 4	\$nolist ; Turn listing off for include file
		=1 52	; End of include file
		53	
002B		54	rseg at 28H
		55	
002B		56	HIGH_TIME: dsw 1
002A		57	LOW_TIME: dsw 1
002C		58	PERIOD: dsw 1
002E		59	HI_EDGE: dsw 1
0030		60	LO_EDGE: dsw 1
		61	
		62	
		63	
20B0		64	cseg at 20B0H
		65	
		66	
20B0 A100011B		67	LD SP, #100H
20B4 B10115		68	LDB IOCO, #00000001B ; Enable HSI 0
20B7 B10F03		69	LDB HSI_MODE, #00001111B ; HSI 0 look for either edge
		70	
20BA 442A2B2C		71	wait: ADD PERIOD, HIGH_TIME, LOW_TIME
20BE 3E1603		72	JBS IOSI, 6, contin ; If FIFO is full
2091 3716F6		73	JBC IOSI, 7, wait ; Wait while no pulse is entered
		74	
2094 B0061C		75	contin: LDB AL, HSI_STATUS ; Load status; Note that reading
		76	MOVX ; HSI_TIME clears HSI_STATUS
		77	
2097 A00420		78	LDB BX, HSI_TIME ; Load the HSI_TIME
		79	
209A 391C09		80	JBS AL, 1, hsi_hi ; Jump if HSI.0 is high
		81	
209D C03020		82	hsi_lo: ST BX, LO_EDGE
20A0 4B2E302B		83	STB HIGH_TIME, LO_EDGE, HI_EDGE
20A4 27E4E8		84	wait
		85	
		86	
20A6 C02E20		87	hsi_hi: ST BX, HI_EDGE

270061-75

SERIES-III MCS-96 MACRO ASSEMBLER, V1.0

SOURCE FILE F3 ENHSI A96

OBJECT FILE F3 ENHSI OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND NOSB

ERR LOC	OBJECT	LINE	SOURCE STATEMENT
		1	\$TITLE ('ENHSI A96 ENHANCED HSI PULSE ROUTINE')
		2	
		3	\$INCLUDE(DEMO96 INC)
		4	\$nolist ; Turn listing off for include file
		52	; End of include file
		53	
002B		54	RSEG AT 28H
		55	
002B		56	TIME DSW 1
002A		57	LAST_RISE DSW 1
002C		58	LAST_FALL DSW 1
002E		59	HSI_SO DSB 1
002F		60	IOS1_BAK DSB 1
0030		61	PERIOD DSW 1
0032		62	LOW_TIME DSW 1
0034		63	HIGH_TIME DSW 1
0036		64	COUNT DSW 1
		65	
2080		66	cseg at 2080H
		67	
2080 A100011B		68	init: LD SP, #100H
		69	
2084 B12516		70	LDB IOC1, #00100101B ; Disable HSD. 4, HSD. 5, HSI_INT=first,
		71	; Enable PWM, TXD, TIMER1_OVRFLOW_INT
		72	
2087 B19903		73	LDB HSI_MODE, #10011001B ; set hsi.1 -, hsi.0 +
208A B10715		74	LDB IOC0, #00000111B ; Enable hsi 0,1
		75	; T2 CLOCK=T2CLK, T2RST=T2RST
		76	; Clear timer2
		77	
208D 717F2F		79	wait: ANDB IOS1_BAK, #01111111B ; Clear IOS1_BAK. 7
2090 90162F		80	ORB IOS1_BAK, IOS1 ; Store into temp to avoid clearing
		81	; other flags which may be needed
2093 372FF7		82	JBC IOS1_BAK, 7, wait ; If hsi is not triggered then
		83	; jump to wait
		84	
2096 5155062E		85	ANDB HSI_SO, HSI_STATUS, #01010101B
209A A0042B		86	LD TIME, HSI_TIME
		87	
		88	END

270061-77

19-115

270061-78

SERIES-III MCS-96 MACRO ASSEMBLER, V1.0

SOURCE FILE: :F3:HSODRV.A96

OBJECT FILE: :F3:HSODRV.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: NOSB

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
			1	%TITLE('HSODRV.A96: Driver module for HSO PWM program')
			2	
			3	HSODRV MODULE MAIN, STACKSIZE(8)
			4	
			5	
			6	PUBLIC HSO_ON_0, HSO_OFF_0
			7	PUBLIC HSO_ON_1, HSO_OFF_1
			8	PUBLIC HSO_TIME, HSO_COMMAND
			9	PUBLIC SP, TIMER1, IOSO
			10	
			11	%INCLUDE(DEMO96.INC)
			12	%nolist ; Turn listing off for include file
			13	% ; End of include file
			14	
			15	
			16	
			17	
			18	
			19	
			20	
			21	
			22	
			23	
			24	
			25	
			26	
			27	
			28	
			29	
			30	
			31	
			32	
			33	
			34	
			35	
			36	
			37	
			38	
			39	
			40	
			41	
			42	
			43	
			44	
			45	
			46	
			47	
			48	
			49	
			50	
			51	
			52	
			53	
			54	
			55	
			56	
			57	
			58	
			59	
			60	
			61	
			62	
			63	
			64	
			65	
			66	
			67	
			68	
			69	
			70	
			71	
			72	
			73	
			74	
			75	
			76	
			77	
			78	
			79	
			80	
			81	
			82	
			83	
			84	
			85	
			86	
			87	
			88	
			89	
			90	
			91	
			92	
			93	
			94	
			95	
			96	
			97	
			98	
			99	
			100	

270061-79

A.6. PWM Using the HSO

19-116

```

209B C02B1C      88      ST      AX, HSO_ON_0
209E C02A20      89      ST      BX, HSO_OFF_0
20A1 0B011C      91      SHR     AX, #1
20A4 0B0120      92      SHR     BX, #1
20A7 C02C1C      93      ST      AX, HSO_ON_1
20AA C02E20      94      ST      BX, HSO_OFF_1
20AD EF0000      95      CALL    wait
20B0 0722      96      INC     CX
20B2 89000F22    97      CMP     CX, #00F00H
20B6 D7DB      98      BNE     loop
20B8 27D2      99      BR      initial
20BA          100     END
          101
          102
          103
          104
          105
          106
          107
          108
          109
          110
          111
          112
          113
          114
          115
          116
          117
          118
          119
          120
          121
          122
          123
          124
          125
          126
          127
          128
          129
          130
          131
          132
          133
          134
          135
          136
          137
          138
          139
          140
          141
          142
          143
          144
          145
          146
          147
          148
          149
          150
          151
          152
          153
          154
          155
          156
          157
          158
          159
          160
          161
          162
          163
          164
          165
          166
          167
          168
          169
          170
          171
          172
          173
          174
          175
          176
          177
          178
          179
          180
          181
          182
          183
          184
          185
          186
          187
          188
          189
          190
          191
          192
          193
          194
          195
          196
          197
          198
          199
          200
          201
          202
          203
          204
          205
          206
          207
          208
          209
          210
          211
          212
          213
          214
          215
          216
          217
          218
          219
          220
          221
          222
          223
          224
          225
          226
          227
          228
          229
          230
          231
          232
          233
          234
          235
          236
          237
          238
          239
          240
          241
          242
          243
          244
          245
          246
          247
          248
          249
          250
          251
          252
          253
          254
          255
          256
          257
          258
          259
          260
          261
          262
          263
          264
          265
          266
          267
          268
          269
          270
          271
          272
          273
          274
          275
          276
          277
          278
          279
          280
          281
          282
          283
          284
          285
          286
          287
          288
          289
          290
          291
          292
          293
          294
          295
          296
          297
          298
          299
          300
          301
          302
          303
          304
          305
          306
          307
          308
          309
          310
          311
          312
          313
          314
          315
          316
          317
          318
          319
          320
          321
          322
          323
          324
          325
          326
          327
          328
          329
          330
          331
          332
          333
          334
          335
          336
          337
          338
          339
          340
          341
          342
          343
          344
          345
          346
          347
          348
          349
          350
          351
          352
          353
          354
          355
          356
          357
          358
          359
          360
          361
          362
          363
          364
          365
          366
          367
          368
          369
          370
          371
          372
          373
          374
          375
          376
          377
          378
          379
          380
          381
          382
          383
          384
          385
          386
          387
          388
          389
          390
          391
          392
          393
          394
          395
          396
          397
          398
          399
          400
          401
          402
          403
          404
          405
          406
          407
          408
          409
          410
          411
          412
          413
          414
          415
          416
          417
          418
          419
          420
          421
          422
          423
          424
          425
          426
          427
          428
          429
          430
          431
          432
          433
          434
          435
          436
          437
          438
          439
          440
          441
          442
          443
          444
          445
          446
          447
          448
          449
          450
          451
          452
          453
          454
          455
          456
          457
          458
          459
          460
          461
          462
          463
          464
          465
          466
          467
          468
          469
          470
          471
          472
          473
          474
          475
          476
          477
          478
          479
          480
          481
          482
          483
          484
          485
          486
          487
          488
          489
          490
          491
          492
          493
          494
          495
          496
          497
          498
          499
          500
          501
          502
          503
          504
          505
          506
          507
          508
          509
          510
          511
          512
          513
          514
          515
          516
          517
          518
          519
          520
          521
          522
          523
          524
          525
          526
          527
          528
          529
          530
          531
          532
          533
          534
          535
          536
          537
          538
          539
          540
          541
          542
          543
          544
          545
          546
          547
          548
          549
          550
          551
          552
          553
          554
          555
          556
          557
          558
          559
          560
          561
          562
          563
          564
          565
          566
          567
          568
          569
          570
          571
          572
          573
          574
          575
          576
          577
          578
          579
          580
          581
          582
          583
          584
          585
          586
          587
          588
          589
          590
          591
          592
          593
          594
          595
          596
          597
          598
          599
          600
          601
          602
          603
          604
          605
          606
          607
          608
          609
          610
          611
          612
          613
          614
          615
          616
          617
          618
          619
          620
          621
          622
          623
          624
          625
          626
          627
          628
          629
          630
          631
          632
          633
          634
          635
          636
          637
          638
          639
          640
          641
          642
          643
          644
          645
          646
          647
          648
          649
          650
          651
          652
          653
          654
          655
          656
          657
          658
          659
          660
          661
          662
          663
          664
          665
          666
          667
          668
          669
          670
          671
          672
          673
          674
          675
          676
          677
          678
          679
          680
          681
          682
          683
          684
          685
          686
          687
          688
          689
          690
          691
          692
          693
          694
          695
          696
          697
          698
          699
          700
          701
          702
          703
          704
          705
          706
          707
          708
          709
          710
          711
          712
          713
          714
          715
          716
          717
          718
          719
          720
          721
          722
          723
          724
          725
          726
          727
          728
          729
          730
          731
          732
          733
          734
          735
          736
          737
          738
          739
          740
          741
          742
          743
          744
          745
          746
          747
          748
          749
          750
          751
          752
          753
          754
          755
          756
          757
          758
          759
          760
          761
          762
          763
          764
          765
          766
          767
          768
          769
          770
          771
          772
          773
          774
          775
          776
          777
          778
          779
          780
          781
          782
          783
          784
          785
          786
          787
          788
          789
          790
          791
          792
          793
          794
          795
          796
          797
          798
          799
          800
          801
          802
          803
          804
          805
          806
          807
          808
          809
          810
          811
          812
          813
          814
          815
          816
          817
          818
          819
          820
          821
          822
          823
          824
          825
          826
          827
          828
          829
          830
          831
          832
          833
          834
          835
          836
          837
          838
          839
          840
          841
          842
          843
          844
          845
          846
          847
          848
          849
          850
          851
          852
          853
          854
          855
          856
          857
          858
          859
          860
          861
          862
          863
          864
          865
          866
          867
          868
          869
          870
          871
          872
          873
          874
          875
          876
          877
          878
          879
          880
          881
          882
          883
          884
          885
          886
          887
          888
          889
          890
          891
          892
          893
          894
          895
          896
          897
          898
          899
          900
          901
          902
          903
          904
          905
          906
          907
          908
          909
          910
          911
          912
          913
          914
          915
          916
          917
          918
          919
          920
          921
          922
          923
          924
          925
          926
          927
          928
          929
          930
          931
          932
          933
          934
          935
          936
          937
          938
          939
          940
          941
          942
          943
          944
          945
          946
          947
          948
          949
          950
          951
          952
          953
          954
          955
          956
          957
          958
          959
          960
          961
          962
          963
          964
          965
          966
          967
          968
          969
          970
          971
          972
          973
          974
          975
          976
          977
          978
          979
          980
          981
          982
          983
          984
          985
          986
          987
          988
          989
          990
          991
          992
          993
          994
          995
          996
          997
          998
          999
          1000
          1001
          1002
          1003
          1004
          1005
          1006
          1007
          1008
          1009
          1010
          1011
          1012
          1013
          1014
          1015
          1016
          1017
          1018
          1019
          1020
          1021
          1022
          1023
          1024
          1025
          1026
          1027
          1028
          1029
          1030
          1031
          1032
          1033
          1034
          1035
          1036
          1037
          1038
          1039
          1040
          1041
          1042
          1043
          1044
          1045
          1046
          1047
          1048
          1049
          1050
          1051
          1052
          1053
          1054
          1055
          1056
          1057
          1058
          1059
          1060
          1061
          1062
          1063
          1064
          1065
          1066
          1067
          1068
          1069
          1070
          1071
          1072
          1073
          1074
          1075
          1076
          1077
          1078
          1079
          1080
          1081
          1082
          1083
          1084
          1085
          1086
          1087
          1088
          1089
          1090
          1091
          1092
          1093
          1094
          1095
          1096
          1097
          1098
          1099
          1100
          1101
          1102
          1103
          1104
          1105
          1106
          1107
          1108
          1109
          1110
          1111
          1112
          1113
          1114
          1115
          1116
          1117
          1118
          1119
          1120
          1121
          1122
          1123
          1124
          1125
          1126
          1127
          1128
          1129
          1130
          1131
          1132
          1133
          1134
          1135
          1136
          1137
          1138
          1139
          1140
          1141
          1142
          1143
          1144
          1145
          1146
          1147
          1148
          1149
          1150
          1151
          1152
          1153
          1154
          1155
          1156
          1157
          1158
          1159
          1160
          1161
          1162
          1163
          1164
          1165
          1166
          1167
          1168
          1169
          1170
          1171
          1172
          1173
          1174
          1175
          1176
          1177
          1178
          1179
          1180
          1181
          1182
          1183
          1184
          1185
          1186
          1187
          1188
          1189
          1190
          1191
          1192
          1193
          1194
          1195
          1196
          1197
          1198
          1199
          1200
          1201
          1202
          1203
          1204
          1205
          1206
          1207
          1208
          1209
          1210
          1211
          1212
          1213
          1214
          1215
          1216
          1217
          1218
          1219
          1220
          1221
          1222
          1223
          1224
          1225
          1226
          1227
          1228
          1229
          1230
          1231
          1232
          1233
          1234
          1235
          1236
          1237
          1238
          1239
          1240
          1241
          1242
          1243
          1244
          1245
          1246
          1247
          1248
          1249
          1250
          1251
          1252
          1253
          1254
          1255
          1256
          1257
          1258
          1259
          1260
          1261
          1262
          1263
          1264
          1265
          1266
          1267
          1268
          1269
          1270
          1271
          1272
          1273
          1274
          1275
          1276
          1277
          1278
          1279
          1280
          1281
          1282
          1283
          1284
          1285
          1286
          1287
          1288
          1289
          1290
          1291
          1292
          1293
          1294
          1295
          1296
          1297
          1298
          1299
          1300
          1301
          1302
          1303
          1304
          1305
          1306
          1307
          1308
          1309
          1310
          1311
          1312
          1313
          1314
          1315
          1316
          1317
          1318
          1319
          1320
          1321
          1322
          1323
          1324
          1325
          1326
          1327
          1328
          1329
          1330
          1331
          1332
          1333
          1334
          1335
          1336
          1337
          1338
          1339
          1340
          1341
          1342
          1343
          1344
          1345
          1346
          1347
          1348
          1349
          1350
          1351
          1352
          1353
          1354
          1355
          1356
          1357
          1358
          1359
          1360
          1361
          1362
          1363
          1364
          1365
          1366
          1367
          1368
          1369
          1370
          1371
          1372
          1373
          1374
          1375
          1376
          1377
          1378
          1379
          1380
          1381
          1382
          1383
          1384
          1385
          1386
          1387
          1388
          1389
          1390
          1391
          1392
          1393
          1394
          1395
          1396
          1397
          1398
          1399
          1400
          1401
          1402
          1403
          1404
          1405
          1406
          1407
          1408
          1409
          1410
          1411
          1412
          1413
          1414
          1415
          1416
          1417
          1418
          1419
          1420
          1421
          1422
          1423
          1424
          1425
          1426
          1427
          1428
          1429
          1430
          1431
          1432
          1433
          1434
          1435
          1436
          1437
          1438
          1439
          1440
          1441
          1442
          1443
          1444
          1445
          1446
          1447
          1448
          1449
          1450
          1451
          1452
          1453
          1454
          1455
          1456
          1457
          1458
          1459
          1460
          1461
          1462
          1463
          1464
          1465
          1466
          1467
          1468
          1469
          1470
          1471
          1472
          1473
          1474
          1475
          1476
          1477
          1478
          1479
          1480
          1481
          1482
          1483
          1484
          1485
          1486
          1487
          1488
          1489
          1490
          1491
          1492
          1493
          1494
          1495
          1496
          1497
          1498
          1499
          1500
          1501
          1502
          1503
          1504
          1505
          1506
          1507
          1508
          1509
          1510
          1511
          1512
          1513
          1514
          1515
          1516
          1517
          1518
          1519
          1520
          1521
          1522
          1523
          1524
          1525
          1526
          1527
          1528
          1529
          1530
          1531
          1532
          1533
          1534
          1535
          1536
          1537
          1538
          1539
          1540
          1541
          1542
          1543
          1544
          1545
          1546
          1547
          1548
          1549
          1550
          1551
          1552
          1553
          1554
          1555
          1556
          1557
          1558
          1559
          1560
          1561
          1562
          1563
          1564
          1565
          1566
          1567
          1568
          1569
          1570
          1571
          1572
          1573
          1574
          1575
          1576
          1577
          1578
          1579
          1580
          1581
          1582
          1583
          1584
          1585
          1586
          1587
          1588
          1589
          1590
          1591
          1592
          1593
          1594
          1595
          1596
          1597
          1598
          1599
          1600
          1601
          1602
          1603
          1604
          1605
          1606
          1607
          1608
          1609
          1610
          1611
          1612
          1613
          1614
          1615
          1616
          1617
          1618
          1619
          1620
          1621
          1622
          1623
          1624
          1625
          1626
          1627
          1628
          1629
          1630
          1631
          1632
          1633
          1634
          1635
          1636
          1637
          1638
          1639
          1640
          1641
          1642
          1643
          1644
          1645
          1646
          1647
          1648
          1649
          1650
          1651
          1652
          1653
          1654
          1655
          1656
          1657
          1658
          1659
          1660
          1661
          1662
          1663
          1664
          1665
          1666
          1667
          1668
          1669
          1670
          1671
          1672
          1673
          1674
          1675
          1676
          1677
          1678
          1679
          1680
          1681
          1682
          1683
          1684
          1685
          1686
          1687
          1688
          1689
          1690
          1691
          1692
          1693
          1694
          1695
          1696
          1697
          1698
          1699
          1700
          1701
          1702
          1703
          1704
          1705
          1706
          1707
          1708
          1709
          1710
          1711
          1712
          1713
          1714
          1715
          1716
          1717
          1718
          1719
          1720
          1721
          1722
          1723
          1724
          1725
          1726
          1727
          1728
          1729
          1730
          1731
          1732
          1733
          1734
          1735
          1736
          1737
          1738
          1739
          1740
          1741
          1742
          1743
          1744
          1745
          1746
          1747
          1748
          1749
          1750
          1751
          1752
          1753
          1754
          1755
          1756
          1757
          1758
          1759
          1760
          1761
          1762
          1763
          1764
          1765
          1766
          1767
          1768
          1769
          1770
          1771
          1772
          1773
          1774
          1775
          1776
          1777
          1778
          1779
          1780
          1781
          1782
          1783
          1784
          1785
          1786
          1787
          1788
          1789
          1790
          1791
          1792
          1793
          1794
          1795
          1796
          1797
          1798
          1799
          1800
          1801
          1802
          1803
          1804
          1805
          1806
          1807
          1808
          1809
          1810
          1811
          1812
          1813
          1814
          1815
          1816
          1817
          1818
          1819
          1820
          1821
          1822
          1823
          1824
          1825
          1826
          1827
          1828
          1829
          1830
          1831
          1832
          1833
          1834
          1835
          1836
          1837
          1838
          1839
          1840
          1841
          1842
          1843
          1844
          1845
          1846
          1847
          1848
          1849
          1850
          1851
          1852
          1853
          1854
          1855
          1856
          1857
          1858
          1859
          1860
          1861
          1862
          1863
          1864
          1865
          1866
          1867
          1868
          1869
          1870
          1871
          1872
          1873
          1874
          1875
          1876
          1877
          1878
          1879
          1880
          1881
          1882
          1883
          1884
          1885
          1886
          1887
          1888
          1889
          1890
          1891
          1892
          1893
          1894
          1895
          1896
          1897
          1898
          1899
          1900
          1901
          1902
          1903
          1904
          1905
          1906
          1907
          1908
          1909
          1910
          1911
          1912
          1913
          1914
          1915
          1916
          1917
          1918
          1919
          1920
          1921
          1922
          1923
          1924
          1925
          1926
          1927
          1928
          1929
          1930
          1931
          1932
          1933
          1934
          1935
          1936
          1937
          1938
          1939
          1940
          1941
          19
```


SERIES-III MCS-96 MACRO ASSEMBLER, V1 0

SOURCE FILE: F3.HSOMOD A96

OBJECT FILE: F3.HSOMOD OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: NOSB

```

ERR LOC  OBJECT          LINE      SOURCE STATEMENT
1          $TITLE('HSOMOD A96: 8096 PWM PROGRAM MODIFIED FOR DRIVER')
2          $PAGEWIDTH(130)
3
4          ; This program will provide 3 PWM outputs on HSO pins 0-2
5          ; The input parameters passed to the program are:
6
7          ;           HSO_ON_N    HSO on time for pin N
8          ;           HSO_OFF_N   HSO off time for pin N
9
10         ;           Where: Times are in timer1 cycles
11         ;           N takes values from 0 to 3
12
13         ;
14         ;
15
16         ; NOTE: Use this file to replace the declaration section of
17         ;       the HSO PWM program from "$INCLUDE(DEMO96.INC)" through
18         ;       the line prior to the label "wait". Also change the last
19         ;       branch in the program to a "RET".
20
21         RSEG
22
23         D_STAT:          dsb          1
24         extrn  HSO_ON_0 :word , HSO_OFF_0 :word
25         extrn  HSO_ON_1 :word , HSO_OFF_1 :word
26         extrn  HSO_TIME :word , HSO_COMMAND :byte
27         extrn  TIMER1  :word , IOSO      :byte
28         extrn  SP       :word
29
30         public  OLD_STAT
31         OLD_STAT:        dsb          1
32         NEW_STAT:        dsb          1
33
34         cseg
35         PUBLIC  wait
36
37         wait:  JBS      IOSO, 6, wait      ; Loop until HSO holding register
38               NOP                               ; is empty
39
40               ; For operation with interrupts 'store_stat:' would be the
41               ; entry point of the routine.
42               ; Note that a DI or PUSHF might have to be added.
43
44

```

270061-81

```

0004      45 store_stat:
0004 510F0002 E 46      ANDB NEW_STAT, IOS0, #0FH      ; Store new status of HSO
0008 980201 R 47      CMPB OLD_STAT, NEW_STAT
0008 DFF3    48      JE wait
000D 940201 R 49      XORB OLD_STAT, NEW_STAT
0010      50
0010      51
0010      52 check_0:
0010 300113 R 53      JBC OLD_STAT, 0, check_1      ; Jump if OLD_STAT(0)=NEW_STAT(0)
0013 380209 R 54      JBS NEW_STAT, 0, set_off_0
0016      55
0016      56 set_on_0:
0016 B13000 E 57      LDB HSO_COMMAND, #00110000B      ; Set HSO for timer1, set pin 0
0019 44000000 E 58      ADD HSO_TIME, TIMER1, HSO_OFF_0      ; Time to set pin = Timer1 value
001D 2007    59      BR check_1      ; + Time for pin to be low
001F      60
001F      61 set_off_0:
001F B11000 E 62      LDB HSO_COMMAND, #00010000B      ; Set HSO for timer1, clear pin 0
0022 44000000 E 63      ADD HSO_TIME, TIMER1, HSO_ON_0      ; Time to clear pin = Timer1 value
0026      64      ; + Time for pin to be high
0026      65 check_1:
0026 310113 R 66      JBC OLD_STAT, 1, check_done      ; Jump if OLD_STAT(1)=NEW_STAT(1)
0029 390209 R 67      JBS NEW_STAT, 1, set_off_1
002C      68
002C      69 set_on_1:
002C B13100 E 70      LDB HSO_COMMAND, #00110001B      ; Set HSO for timer1, set pin 1
002F 44000000 E 71      ADD HSO_TIME, TIMER1, HSO_OFF_1      ; Time to set pin = Timer1 value
0033 2007    72      BR check_done      ; + Time for pin to be high
0035      73
0035      74 set_off_1:
0035 B11100 E 75      LDB HSO_COMMAND, #00010001B      ; Set HSO for timer1, clear pin 1
0038 44000000 E 76      ADD HSO_TIME, TIMER1, HSO_ON_1      ; Time to clear pin = Timer1 value
003C      77      ; + Time for pin to be high
003C      78 check_done:
003C 800201 R 79      LDB OLD_STAT, NEW_STAT      ; Store current status and
003F F0      80      CHB      ; wait for interrupt flag
003F F0      81
003F F0      82 RET
0040      83      use "BR wait" if this routine is used with the driver
0040      84
0040      85 END

```

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

270061-82

SERIES-III MCS-96 MACRO ASSEMBLER, V1 0

SOURCE FILE F3 SP A96

OBJECT FILE F3 SP OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND NOSB

ERR LOC	OBJECT	LINE	SOURCE STATEMENT
		1	
		2	\$TITLE('SP.A96: SERIAL PORT DEMO PROGRAM')
		3	
		4	\$INCLUDE(DEMO96.INC)
		5	\$nolist END Turn listing off for include file
		53	; End of include file
		54	
0028	LD	55	rseg at 28H
		56	
0028		57	CHR: dsb 1
0029	MOV	58	SPTMP: dsb 1
002A		59	TEMP0: dsb 1
002B		60	TEMP1: dsb 1
002C		61	RCV_FLAG: dsb 1
200C		62	
		63	cseg at 200CH
		64	
200C	9C20	65	DCW ser_port_int
		66	
2080		67	cseg at 2080H
		68	
2080	A100011B	69	LD SP, #100H
		70	
2084	B12016	71	LDB OF IDCI, #00100000B
		72	
		73	; Baud rate = input frequency / (64*baud_val)
		74	H20 baud_val = (input frequency/64) / baud rate
		75	
		76	
0027		77	baud_val equ 39 ; 39 = (12,000,000/64)/4800 baud
		78	
0080		79	BAUD_HIGH equ ((baud_val-1)/256) OR 80H ; Set MSB to 1
0026		80	BAUD_LOW equ (baud_val-1) MOD 256
		81	
		82	
2087	B1260E	83	LDB BAUD_REG, #BAUD_LOW
208A	B1800E	84	LDB BAUD_REG, #BAUD_HIGH
		85	
		86	
		87	
		88	
		89	
		90	
		91	
		92	
		93	
		94	
		95	
		96	
		97	
		98	
		99	
		100	

270061-83

A.7. Serial Port (Continued)

```

208D B14911      86          LDB      SPCON, #01001001B      ; Enable receiver, Mode 1
2090 C42807      87
2093 B1202A      88          ; The serial port is now initialized
2096 B1400B      89
2099 FB          90
209A 27FE        91          STB      SBUF, CHR      ; Clear serial Port
209D B01129      92          LDB      TEMPO, #00100000B      ; Set TI-temp
20A0 90292A      93
20A3 716029      94          LDB      INT_MASK, #01000000B      ; Enable Serial Port Interrupt
20A6 D7F5        95          EI
20AB 362A09      96          loop: BR      loop      ; Wait for serial port interrupt
20AB C42807      97
20AE 71BF2A      98          ser_port_int:
20B1 B1FF2C      99          PUSHF
20B4 302C18      100         rd_again:
20B7 352A15      101         LDB      SPTEMP, SPSTAT      ; This section of code can be replaced
20BA B02807      102         ORB      TEMPO, SP_STAT      ; with "ORB TEMPO, SP_STAT" when the
20BD 71DF2A      103         ORB      TEMPO, SPTEMP      ; serial port TI and RI bugs are fixed
20C0 717F2B      104         ANDB     SPTEMP, #01100000B
20C3 990D2B      105         JNE      rd_again      ; Repeat until TI and RI are properly cleared
20C6 D705        106
20C8 B10A2B      107         get_byte:
20CB 2002        108         JBC      TEMPO, 6, put_byte      ; If RI-temp is not set
20CD 112C        109         STB      SBUF, CHR      ; Store byte
20CF F3          110         ANDB     TEMPO, #10111111B      ; CLR RI-temp
20D0 F0          111         LDB      RCV_FLAG, #OFFH      ; Set bit-received flag
20D1          112
20D1          113         put_byte:
20D1          114         JBC      RCV_FLAG, 0, continue      ; If receive flag is cleared
20D1          115         JBC      TEMPO, 5, continue      ; If TI was not set
20D1          116         LDB      SBUF, CHR      ; Send byte
20D1          117         ANDB     TEMPO, #11011111B      ; CLR TI-temp
20D1          118         ANDB     CHR, #01111111B      ; This section of code appends
20D1          119         CMPB     CHR, #0DH      ; an LF after a CR is sent
20D1          120         JNE      clr_rcv
20D1          121         LDB      CHR, #0AH
20D1          122         BR      continue
20D1          123
20D1          124         clr_rcv:
20D1          125         CLR      RCV_FLAG      ; Clear bit-received flag
20D1          126
20D1          127         continue:
20D1          128         POPF
20D1          129         RET
20D1          130
20D1          131         END
20D1          132

```

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

270061-84

SERIES-III MCS-96 MACRO ASSEMBLER, V1 0

SOURCE FILE: F3:ATOD.A96
OBJECT FILE: F3:ATOD.OBJ
CONTROLS SPECIFIED IN INVOCATION COMMAND: NOSB

ERR LOC	OBJECT	LINE	SOURCE STATEMENT
		1	\$TITLE('ATOD.A96: SCANNING THE A TO D CHANNELS')
		2	
		3	\$INCLUDE(DEMO96.INC)
		4	\$nolist ; Turn listing off for include file
		52	End of include file
		53	
	0028	54	RSEG at 28H
		55	
	0020	56	BL EQU BX:BYTE
	001E	57	DL EQU DX:BYTE
		58	
	0028	59	RESULT_TABLE:
	0028	60	RESULT_1: dsw 1
	002A	61	RESULT_2: dsw 1
	002C	62	RESULT_3: dsw 1
	002E	63	RESULT_4: dsw 1
		64	
	2080	65	
		66	cseg at 2080H
		67	
	2080	68	LD SP, #100H ; Set Stack Pointer
	2084	69	CLR BX
		70	
	2086	71	next: ADDB AD_COMMAND, BL, #1000B ; Start conversion on channel
		72	indicated by BL register
	208A	73	
	208B	74	NDP ; Wait for conversion to start
	208C	75	NDP
	3802FD	76	check: JBS AD_RESULT_LO, 3, check ; Wait while A to D is busy
		77	
	208F	78	LDB AL, AD_RESULT_LO ; Load low order result
	2092	79	LDB AH, AD_RESULT_HI ; Load high order result
		80	
	2095	81	ADDB DL, BL, BL ; DL=BL*2
	2099	82	LDBZE DX, DL
	209C	83	ST AX, RESULT_TABLE[DX] ; Store result indexed by BL*2
		84	
	20A0	85	INCB BL ; Increment BL modulo 4
	1720	86	

270061-85

APPENDIX B HSO AND A TO D UNDER INTERRUPT CONTROL

SERIES-III MCS-96 MACRO ASSEMBLER, V1.0

SOURCE FILE: :F3:A2DHSO.A96

OBJECT FILE: :F3:A2DHSO.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: NOSB

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
			1	\$TITLE ('A2DHSO.A96: GENERATING PWM OUTPUTS FROM A TO D INPUTS')
			2	
			3	; This program will provide 3 PWM outputs on HSO pins 0-2
			4	; and one on the PWM.
			5	
			6	; The PWM values are determined by the input to the A/D converter.
			7	
			8	;;;
			9	
			10	\$INCLUDE(DEMO96.INC)
=1			11	\$nolist ; Turn listing off for include file
=1			59	; End of include file
			60	
	002B		61	RSEG AT 28H.
			62	
	001E		63	DL EQU DX:BYTE
			64	
	002B		65	ON_TIME:
	002B		66	PWM_TIME_1: DSW 1
	002A		67	HSO_ON_0: DSW 1
	002C		68	HSO_ON_1: DSW 1
	002E		69	HSO_ON_2: DSW 1
			70	
	0030		71	RESULT_TABLE:
	0030		72	RESULT_0: DSW 1
	0032		73	RESULT_1: DSW 1
	0034		74	RESULT_2: DSW 1
	0036		75	RESULT_3: DSW 1
			76	
	003B		77	NXT_ON_T: DSW 1
	003A		78	NXT_OFF_0: DSW 1
	003C		79	NXT_OFF_1: DSW 1
	003E		80	NXT_OFF_2: DSW 1
	0040		81	COUNT: DSL 1
	0044		82	AD_NUM: DSW 1 Channel being converted
	0046		83	TMP: DSW 1
	0048		84	HSO_PER: DSW 1
	004A		85	LAST_LOAD: DSW 1
			86	

270061-87

```

2000      87  cseg  AT 2000H
2000 8020      89      DCW      start      ; Timer_ovf_int
2002 1D21      90      DCW      Atod_done_int
2004 8020      91      DCW      start      ; HSI_data_int
2006 CC20      92      DCW      HSO_exec_int
          93
          94  $EJECT
          95
2080      96  cseg  AT 2080H
          97
2080 A100011B    98  start: LD      SP, #100H      ; Set Stack Pointer
2084 011C      99      CLR     AX
2086 051C     100  wait: DEC     AX      ; wait approx. 0.2 seconds for
2088 D7FC      101      JNE     wait      ; SBE to finish communications
          102
208A 1144      103      CLR     AD_NUM
          104
208C A180002B    105      LD      PWM_TIME_1, #080H
2090 A100014B    106      LD      HSO_PER, #100H
2094 A140002A    107      LD      HSO_ON_0, #040H
2098 A180002C    108      LD      HSO_ON_1, #080H
209C A1C0002E    109      LD      HSO_ON_2, #0C0H
          110
20A0 4500010A3B  111      ADD     NXT_ON_T, Timer1, #100H
          112
20A5 B13606     113      LDB     HSO_COMMAND, #00110110B ; Set HSO for timer1, set pin 0.1
20AB A03804     114      LD      HSO_TIME, NXT_ON_T ; with interrupt
20AD FD        115      NOP
20AC FD        116      NOP
20AD B12206     117      LDB     HSO_COMMAND, #00100010B ; Set HSO for timer1, set pin 2
20B0 643804     118      ADD     HSO_TIME, NXT_ON_T ; without interrupt
          119
20B3 91074A     120      ORB     LAST_LOAD, #00000111B ; Last loaded value was set all pins
20B6 B10A08     121      LDB     INT_MASK, #00001010B ; Enable HSO and A/D interrupts
20B9 B10A09     122      LDB     INT_PENDING, #00001010B ; Fake an A/D and HSO interrupt
20BC FB        123      EI
          124
20BD 91010F     125  loop: ORB     Port1, #00000001B ; set P1.0
20C0 65010040    126      ADD     COUNT, #01
20C4 A40042     127      ADDC    COUNT+2, zero
20C7 71FE0F     128      ANDB    Port1, #11111110B ; clear P1.0
20CA 27F1      129      BR      loop
          130
          131  $EJECT
          132
          133

```

270061-88


```

132
133
134 ..... HSO EXECUTED INTERRUPT .....
135 .....
136
137 HSO_exec_int:
20CC F2 138 PUSHF
20CD 91020F 139 ORB Port1, #00000010B ; Set pl.1
140
141 SUB TMP, TIMER1, NXT_ON_T
20D0 48380A46 141 SUB TMP, TIMER1, NXT_ON_T
20D4 880046 142 CMP TMP, ZERO
20D7 DE19 143 JLT set_off_times
144
145 set_off_times:
20D9 81010E 145 set_off_times:
20D9 64483B 146 ADD NXT_ON_T, HSO_PER
20DC B13606 147 LDB HSO_COMMAND, #00110110B ; Set HSO for timer1, set pin 0.1
20DF A03804 148 LD HSO_TIME, NXT_ON_T
20E2 FD 149 NOP
20E3 FD 150 NOP
20E4 B12206 151 LDB HSO_COMMAND, #00100010B ; Set HSO for timer1, set pin 2
20E7 A03804 152 LD HSO_TIME, NXT_ON_T
153
154 ORB LAST_LOAD, #00000111B ; Last loaded value was all ones
155
156 LDB PWM_CONTROL, PWM_TIME_1 ; Now is as good a time as any
157 ; to update the PWM reg
20ED 802817 156 LDB PWM_CONTROL, PWM_TIME_1
20F0 2026 158 BR check_done
159
160
161 set_off_times:
20F2 304A23 162 JBC LAST_LOAD, 0, check_done
163
164 ADD NXT_OFF_0, NXT_ON_T, HSO_ON_0
20F5 442A383A 164 ADD NXT_OFF_0, NXT_ON_T, HSO_ON_0
20F9 B11006 165 LDB HSO_COMMAND, #00010000B ; Set HSO for timer1, clear pin 0
20FC A03A04 166 LD HSO_TIME, NXT_OFF_0
167
168 NOP
20FF FD 168 NOP
169
170 ADD NXT_OFF_1, NXT_ON_T, HSO_ON_1
2100 442C383C 169 ADD NXT_OFF_1, NXT_ON_T, HSO_ON_1
2104 B11106 170 LDB HSO_COMMAND, #00010001B ; Set HSO for timer1, clear pin 1
2107 A03C04 171 LD HSO_TIME, NXT_OFF_1
172
173 NOP
210A FD 173 NOP
174
175 ADD NXT_OFF_2, NXT_ON_T, HSO_ON_2
210B 442E383E 174 ADD NXT_OFF_2, NXT_ON_T, HSO_ON_2
210F B11206 175 LDB HSO_COMMAND, #00010010B ; Set HSO for timer1, clear pin 2
2112 A03E04 176 LD HSO_TIME, NXT_OFF_2
177
178 ANDB LAST_LOAD, #11111000B ; Last loaded value was all 0s
2115 71FB4A 178 ANDB LAST_LOAD, #11111000B
179
180 check_done:
2118 71D5F 180
2118 71FDOF 181 ANDB Port1, #11111101B ; Clear Pl.1
182
5000

```

```

211B F3      182      POPF
211C F0      183      RET
184
185      $EJECT
186
187      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
188      ; A TO D COMPLETE INTERRUPT ;;;;;;;;;;;;;;;;;;;;;;;;;;
189      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
190
191      ATOD_done_int:
192      PUSHF
193      ORB     Port1, #00000100B      ; Set P1.2
194
195      ANDB    AL, AD_RESULT_LO, #11000000B      ; Load low order result
196      LDB     AH, AD_RESULT_HI      ; Load high order result
197      ADDB    DL, AD_NUM, AD_NUM      ; DL= AD_NUM *2
198      LDBZE   DX, DL
199      ST      AX, RESULT_TABLE[DX]      ; Store result indexed by DX
200
201      CMPB    AL, #01000000B      ; Round up if needed
202      JNH     no_rnd
203      CMPB    AH, #0FFH      ; Don't increment if AH=0FFH
204      JE      no_rnd
205      INCB    AH
206
207      no_rnd: LDB     AL, AH      ; Align byte and change to word
208      CLRB    AH
209      ST      AX, ON_TIME[DX]
210
211      INCB    AD_NUM
212      ANDB    AD_NUM, #03H      ; Keep AD_NUM between 0 and 3
213
214      next:  ADDB    AD_COMMAND, AD_NUM, #1000B      ; Start conversion on channel
215      ; indicated by AD_NUM register
216      ANDB    Port1, #1111011B      ; Clear P1.2
217      POPF
218      RET
219
220      END
221
2156

```

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

270061-90

APPENDIX C SOFTWARE SERIAL PORT

SERIES-III MCS-96 MACRO ASSEMBLER, V1 0

SOURCE FILE: F3:SWPORT.A96

OBJECT FILE: F3:SWPORT.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: NOSB

```

ERR LOC OBJECT LINE SOURCE STATEMENT
1 $TITLE('SWPORT.A96 : SOFTWARE IMPLEMENTED ASYNCHRONOUS SERIAL PORT')
2
3 ; This module provides a software implemented asynchronous serial port
4 ; for the 8096. HSD.5 is used for transmit data. HSI.2 is used for
5 ; receive data. Note: the choice of HSD.5 and HSI.2 is arbitrary).
6
7 $INCLUDE(DEMO96.INC)
8 $nolist ; Turn listing off for include file
9 $EEL; End of include file
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
```

```

2080          88          cseg at 2080h
2080          89
2080          90 reset_loc:
2080          91 ; The 8096 starts executing here on reset. the program will initialize the
2080          92 ; the software serial port and run a simple test to exercize it.
2080          93
2080 FA          94          di
2081 A1F00018      95          ld      sp,#0F0h
2085 C9C012      96          push   #4800
208B EF0000      97          call   setup_serial_port
208B B16C0B      98          ldb     int_mask,#01101100b ; serial. swt, hso, hsi
208E FB          99          ei
2080          100
2080          101 ;
208F          102 test1:
2080          103 ; A simple test of the serial port routines.
2080          104 ; While no characters are received an incrementing pattern is sent to the
2080          105 ; serial output. When a character is received the incrementing pattern
2080          106 ; "jumps" to the character received and proceeds from there.
2080          107 ;
2080          108          CR      equ      0DH ; Carriage return
208F B10DOC      R      109          ldb     char,#CR
2092          110 testloop:
2092 AC0C1C      R      111          ldbz   ax,char
2095 C81C          112          push   ax
2097 EF3000      R      113          call   char_out
2090          114
209A 990DOC      R      115          cmpb   char,#CR ; Pause on Carriage return
209D D706          116          bne    nopause
209F 011C          117          clr     ax
20A1          118 pause:
20A1 071C          119          inc     ax
20A3 D7FC          120          bne    pause
20A5          121 nopause:
20A5 170C      R      122
20A7          123          incb   char
20A7 EF4400      R      124 test2:
20AA 98001C      125          call   csts ; char ready?
20AD DFE3          126          cmpb   al,0
20AF EF4C00      127          be     testloop ; loop if not
20B2 B01C0C      R      128          call   char_in
20B5 27DB          129          ldb     char,al
2080          130          br     testloop
2080          131 $eject
2080          132
2080          133

```

270061-92


```

0000      132
133      133      cseg
134      134
0000      135      setup_serial_port.
136      136      ; Called on system reset to initiate the software serial port.
137      137
0000 CC22      138      pop      cx      ; the return address
0002 CC20      139      pop      bx      ; the baud rate (in decimal)
0004 A107001E  140      ld      dx,#0007h      ; dx:ax =500,000 (assumes 12 Mhz crystal)
0008 A120A11C  141      ld      ax,#0A120h
000C BC201C    142      divu     ax,bx      ; calculate the baud count (500,000/baudrate)
000F C00B1C    143      st      ax,baud_count
0012 C00600    R 144      st      0,serial_out      ; clear serial out
0015 B16016    R 145      ldb     ioc1,#01100000b ; Enable HSO_5 and Txd
0018 3E15FD    146      bbs     ios0,b,$      ; Wait for room in the HSO CAM
                                ; and issue a MARK command.
001B 44140A0A  R 147
001F B13506    148      add     txd_time,timer1,20
0022 A00A04    R 149      ldb     hso_command,#mark_command
0025 1102      R 150      ld      hso_time,txd_time
0027 1103      R 151      clrb    rcv_buf      ; clear out the receive variables
0029 1101      R 152      clrb    rcv_reg
002B EF4800    153      clrb    rcv_state
002E E322      154      call   init_receive      ; setup to detect a start bit
                                ; return
0030 E322      155      br      [cx]
                                ;
0030 E322      156      $reject
                                ;
0030 E322      157
0030 E322      158      char_out
0030 E322      159      ; Output character to the software serial port.
0030 E322      160
0030 CC22      161      pop      cx      ; the return address
0032 CC20      162      pop      bx      ; the character for output
0034 B10121    163      ldb     (bx+1),#01h      ; add the start and stop bits
0037 642020    164      add     bx,bx      ; to the char and leave as 16 bit
003A          165      wait_for_xmit:
003A B80006    R 166      cmp     serial_out,0      ; wait for serial_out=0 (it will be cleared by
003D D7FB      167      bne     wait_for_xmit      ; the hso interrupt process)
003F C00620    R 168      st      bx,serial_out      ; put the formatted character in serial_out
0042 E322      169      br      [cx]      ; return to caller
0044          170
0044          171      csts:
0044          172      ; Returns "true" (ax<>0) if char_in has a character.
0044          173      ;
0044 011C      174      clr     ax
0046 300102    R 175      bbc     rcv_state,0,csts_exit
0049 071C      176      inc     ax
004B          177      csts_exit:
004B F0        178      ret
004C          179
004C          180      char_in
004C          181
004C          182
004C          183
004C          184
004C          185
004C          186
004C          187
004C          188
004C          189
004C          190
004C          191
004C          192
004C          193
004C          194
004C          195
004C          196
004C          197
004C          198
004C          199
004C          200
004C          201
004C          202
004C          203
004C          204
004C          205
004C          206
004C          207
004C          208
004C          209
004C          210
004C          211
004C          212
004C          213
004C          214
004C          215
004C          216
004C          217
004C          218
004C          219
004C          220
004C          221
004C          222
004C          223
004C          224
004C          225
004C          226
004C          227
004C          228
004C          229
004C          230
004C          231
004C          232
004C          233
004C          234
004C          235
004C          236
004C          237
004C          238
004C          239
004C          240
004C          241
004C          242
004C          243
004C          244
004C          245
004C          246
004C          247
004C          248
004C          249
004C          250
004C          251
004C          252
004C          253
004C          254
004C          255
004C          256
004C          257
004C          258
004C          259
004C          260
004C          261
004C          262
004C          263
004C          264
004C          265
004C          266
004C          267
004C          268
004C          269
004C          270
004C          271
004C          272
004C          273
004C          274
004C          275
004C          276
004C          277
004C          278
004C          279
004C          280
004C          281
004C          282
004C          283
004C          284
004C          285
004C          286
004C          287
004C          288
004C          289
004C          290
004C          291
004C          292
004C          293
004C          294
004C          295
004C          296
004C          297
004C          298
004C          299
004C          300
004C          301
004C          302
004C          303
004C          304
004C          305
004C          306
004C          307
004C          308
004C          309
004C          310
004C          311
004C          312
004C          313
004C          314
004C          315
004C          316
004C          317
004C          318
004C          319
004C          320
004C          321
004C          322
004C          323
004C          324
004C          325
004C          326
004C          327
004C          328
004C          329
004C          330
004C          331
004C          332
004C          333
004C          334
004C          335
004C          336
004C          337
004C          338
004C          339
004C          340
004C          341
004C          342
004C          343
004C          344
004C          345
004C          346
004C          347
004C          348
004C          349
004C          350
004C          351
004C          352
004C          353
004C          354
004C          355
004C          356
004C          357
004C          358
004C          359
004C          360
004C          361
004C          362
004C          363
004C          364
004C          365
004C          366
004C          367
004C          368
004C          369
004C          370
004C          371
004C          372
004C          373
004C          374
004C          375
004C          376
004C          377
004C          378
004C          379
004C          380
004C          381
004C          382
004C          383
004C          384
004C          385
004C          386
004C          387
004C          388
004C          389
004C          390
004C          391
004C          392
004C          393
004C          394
004C          395
004C          396
004C          397
004C          398
004C          399
004C          400
004C          401
004C          402
004C          403
004C          404
004C          405
004C          406
004C          407
004C          408
004C          409
004C          410
004C          411
004C          412
004C          413
004C          414
004C          415
004C          416
004C          417
004C          418
004C          419
004C          420
004C          421
004C          422
004C          423
004C          424
004C          425
004C          426
004C          427
004C          428
004C          429
004C          430
004C          431
004C          432
004C          433
004C          434
004C          435
004C          436
004C          437
004C          438
004C          439
004C          440
004C          441
004C          442
004C          443
004C          444
004C          445
004C          446
004C          447
004C          448
004C          449
004C          450
004C          451
004C          452
004C          453
004C          454
004C          455
004C          456
004C          457
004C          458
004C          459
004C          460
004C          461
004C          462
004C          463
004C          464
004C          465
004C          466
004C          467
004C          468
004C          469
004C          470
004C          471
004C          472
004C          473
004C          474
004C          475
004C          476
004C          477
004C          478
004C          479
004C          480
004C          481
004C          482
004C          483
004C          484
004C          485
004C          486
004C          487
004C          488
004C          489
004C          490
004C          491
004C          492
004C          493
004C          494
004C          495
004C          496
004C          497
004C          498
004C          499
004C          500
004C          501
004C          502
004C          503
004C          504
004C          505
004C          506
004C          507
004C          508
004C          509
004C          510
004C          511
004C          512
004C          513
004C          514
004C          515
004C          516
004C          517
004C          518
004C          519
004C          520
004C          521
004C          522
004C          523
004C          524
004C          525
004C          526
004C          527
004C          528
004C          529
004C          530
004C          531
004C          532
004C          533
004C          534
004C          535
004C          536
004C          537
004C          538
004C          539
004C          540
004C          541
004C          542
004C          543
004C          544
004C          545
004C          546
004C          547
004C          548
004C          549
004C          550
004C          551
004C          552
004C          553
004C          554
004C          555
004C          556
004C          557
004C          558
004C          559
004C          560
004C          561
004C          562
004C          563
004C          564
004C          565
004C          566
004C          567
004C          568
004C          569
004C          570
004C          571
004C          572
004C          573
004C          574
004C          575
004C          576
004C          577
004C          578
004C          579
004C          580
004C          581
004C          582
004C          583
004C          584
004C          585
004C          586
004C          587
004C          588
004C          589
004C          590
004C          591
004C          592
004C          593
004C          594
004C          595
004C          596
004C          597
004C          598
004C          599
004C          600
004C          601
004C          602
004C          603
004C          604
004C          605
004C          606
004C          607
004C          608
004C          609
004C          610
004C          611
004C          612
004C          613
004C          614
004C          615
004C          616
004C          617
004C          618
004C          619
004C          620
004C          621
004C          622
004C          623
004C          624
004C          625
004C          626
004C          627
004C          628
004C          629
004C          630
004C          631
004C          632
004C          633
004C          634
004C          635
004C          636
004C          637
004C          638
004C          639
004C          640
004C          641
004C          642
004C          643
004C          644
004C          645
004C          646
004C          647
004C          648
004C          649
004C          650
004C          651
004C          652
004C          653
004C          654
004C          655
004C          656
004C          657
004C          658
004C          659
004C          660
004C          661
004C          662
004C          663
004C          664
004C          665
004C          666
004C          667
004C          668
004C          669
004C          670
004C          671
004C          672
004C          673
004C          674
004C          675
004C          676
004C          677
004C          678
004C          679
004C          680
004C          681
004C          682
004C          683
004C          684
004C          685
004C          686
004C          687
004C          688
004C          689
004C          690
004C          691
004C          692
004C          693
004C          694
004C          695
004C          696
004C          697
004C          698
004C          699
004C          700
004C          701
004C          702
004C          703
004C          704
004C          705
004C          706
004C          707
004C          708
004C          709
004C          710
004C          711
004C          712
004C          713
004C          714
004C          715
004C          716
004C          717
004C          718
004C          719
004C          720
004C          721
004C          722
004C          723
004C          724
004C          725
004C          726
004C          727
004C          728
004C          729
004C          730
004C          731
004C          732
004C          733
004C          734
004C          735
004C          736
004C          737
004C          738
004C          739
004C          740
004C          741
004C          742
004C          743
004C          744
004C          745
004C          746
004C          747
004C          748
004C          749
004C          750
004C          751
004C          752
004C          753
004C          754
004C          755
004C          756
004C          757
004C          758
004C          759
004C          760
004C          761
004C          762
004C          763
004C          764
004C          765
004C          766
004C          767
004C          768
004C          769
004C          770
004C          771
004C          772
004C          773
004C          774
004C          775
004C          776
004C          777
004C          778
004C          779
004C          780
004C          781
004C          782
004C          783
004C          784
004C          785
004C          786
004C          787
004C          788
004C          789
004C          790
004C          791
004C          792
004C          793
004C          794
004C          795
004C          796
004C          797
004C          798
004C          799
004C          800
004C          801
004C          802
004C          803
004C          804
004C          805
004C          806
004C          807
004C          808
004C          809
004C          810
004C          811
004C          812
004C          813
004C          814
004C          815
004C          816
004C          817
004C          818
004C          819
004C          820
004C          821
004C          822
004C          823
004C          824
004C          825
004C          826
004C          827
004C          828
004C          829
004C          830
004C          831
004C          832
004C          833
004C          834
004C          835
004C          836
004C          837
004C          838
004C          839
004C          840
004C          841
004C          842
004C          843
004C          844
004C          845
004C          846
004C          847
004C          848
004C          849
004C          850
004C          851
004C          852
004C          853
004C          854
004C          855
004C          856
004C          857
004C          858
004C          859
004C          860
004C          861
004C          862
004C          863
004C          864
004C          865
004C          866
004C          867
004C          868
004C          869
004C          870
004C          871
004C          872
004C          873
004C          874
004C          875
004C          876
004C          877
004C          878
004C          879
004C          880
004C          881
004C          882
004C          883
004C          884
004C          885
004C          886
004C          887
004C          888
004C          889
004C          890
004C          891
004C          892
004C          893
004C          894
004C          895
004C          896
004C          897
004C          898
004C          899
004C          900
004C          901
004C          902
004C          903
004C          904
004C          905
004C          906
004C          907
004C          908
004C          909
004C          910
004C          911
004C          912
004C          913
004C          914
004C          915
004C          916
004C          917
004C          918
004C          919
004C          920
004C          921
004C          922
004C          923
004C          924
004C          925
004C          926
004C          927
004C          928
004C          929
004C          930
004C          931
004C          932
004C          933
004C          934
004C          935
004C          936
004C          937
004C          938
004C          939
004C          940
004C          941
004C          942
004C          943
004C          944
004C          945
004C          946
004C          947
004C          948
004C          949
004C          950
004C          951
004C          952
004C          953
004C          954
004C          955
004C          956
004C          957
004C          958
004C          959
004C          960
004C          961
004C          962
004C          963
004C          964
004C          965
004C          966
004C          967
004C          968
004C          969
004C          970
004C          971
004C          972
004C          973
004C          974
004C          975
004C          976
004C          977
004C          978
004C          979
004C          980
004C          981
004C          982
004C          983
004C          984
004C          985
004C          986
004C          987
004C          988
004C          989
004C          990
004C          991
004C          992
004C          993
004C          994
004C          995
004C          996
004C          997
004C          998
004C          999
004C          1000
004C          1001
004C          1002
004C          1003
004C          1004
004C          1005
004C          1006
004C          1007
004C          1008
004C          1009
004C          1010
004C          1011
004C          1012
004C          1013
004C          1014
004C          1015
004C          1016
004C          1017
004C          1018
004C          1019
004C          1020
004C          1021
004C          1022
004C          1023
004C          1024
004C          1025
004C          1026
004C          1027
004C          1028
004C          1029
004C          1030
004C          1031
004C          1032
004C          1033
004C          1034
004C          1035
004C          1036
004C          1037
004C          1038
004C          1039
004C          1040
004C          1041
004C          1042
004C          1043
004C          1044
004C          1045
004C          1046
004C          1047
004C          1048
004C          1049
004C          1050
004C          1051
004C          1052
004C          1053
004C          1054
004C          1055
004C          1056
004C          1057
004C          1058
004C          1059
004C          1060
004C          1061
004C          1062
004C          1063
004C          1064
004C          1065
004C          1066
004C          1067
004C          1068
004C          1069
004C          1070
004C          1071
004C          1072
004C          1073
004C          1074
004C          1075
004C          1076
004C          1077
004C          1078
004C          1079
004C          1080
004C          1081
004C          1082
004C          1083
004C          1084
004C          1085
004C          1086
004C          1087
004C          1088
004C          1089
004C          1090
004C          1091
004C          1092
004C          1093
004C          1094
004C          1095
004C          1096
004C          1097
004C          1098
004C          1099
004C          1100
004C          1101
004C          1102
004C          1103
004C          1104
004C          1105
004C          1106
004C          1107
004C          1108
004C          1109
004C          1110
004C          1111
004C          1112
004C          1113
004C          1114
004C          1115
004C          1116
004C          1117
004C          1118
004C          1119
004C          1120
004C          1121
004C          1122
004C          1123
004C          1124
004C          1125
004C          1126
004C          1127
004C          1128
004C          1129
004C          1130
004C          1131
004C          1132
004C          1133
004C          1134
004C          1135
004C          1136
004C          1137
004C          1138
004C          1139
004C          1140
004C          1141
004C          1142
004C          1143
004C          1144
004C          1145
004C          1146
004C          1147
004C          1148
004C          1149
004C          1150
004C          1151
004C          1152
004C          1153
004C          1154
004C          1155
004C          1156
004C          1157
004C          1158
004C          1159
004C          1160
004C          1161
004C          1162
004C          1163
004C          1164
004C          1165
004C          1166
004C          1167
004C          1168
004C          1169
004C          1170
004C          1171
004C          1172
004C          1173
004C          1174
004C          1175
004C          1176
004C          1177
004C          1178
004C          1179
004C          1180
004C          1181
004C          1182
004C          1183
004C          1184
004C          1185
004C          1186
004C          1187
004C          1188
004C          1189
004C          1190
004C          1191
004C          1192
004C          1193
004C          1194
004C          1195
004C          1196
004C          1197
004C          1198
004C          1199
004C          1200
004C          1201
004C          1202
004C          1203
004C          1204
004C          1205
004C          1206
004C          1207
004C          1208
004C          1209
004C          1210
004C          1211
004C          1212
004C          1213
004C          1214
004C          1215
004C          1216
004C          1217
004C          1218
004C          1219
004C          1220
004C          1221
004C          1222
004C          1223
004C          1224
004C          1225
004C          1226
004C          1227
004C          1228
004C          1229
004C          1230
004C          1231
004C          1232
004C          1233
004C          1234
004C          1235
004C          1236
004C          1237
004C          1238
004C          1239
004C          1240
004C          1241
004C          1242
004C          1243
004C          1244
004C          1245
004C          1246
004C          1247
004C          1248
004C          1249
004C          1250
004C          1251
004C          1252
004C          1253
004C          1254
004C          1255
004C          1256
004C          1257
004C          1258
004C          1259
004C          1260
004C          1261
004C          1262
004C          1263
004C          1264
004C          1265
004C          1266
004C          1267
004C          1268
004C          1269
004C          1270
```

```

181 ; Get a character from the software serial port
182 ;
183 ; wait for character ready
184 bbc rcve_state,0,char_in
185 pushf ; set up a critical region
186 andb rcve_state,#not(rxdy)
187 ldbze al,rcve_buf
188 popf
189 ret ; leave the critical region
190
191 $eject
192 hso_isr:
193 ; Fields the hso interrupts and performs the serialization of the data.
194 ; Note: this routine would be incorporated into the hso service strategy
195 ; for an actual system.
196
197 cseg at 2006h
198 dcw hso_isr ; Set up vector
199
200 cseg
201 pushf
202 add txd_time,baud_count
203 cmp serial_out,0 ; if character is done send a mark
204 be send_mark
205 shr serial_out,#1 ; else send bit 0 of serial_out and shift
206 bc send_mark ; serial_out left one place.
207
208 send_space:
209 ldb hso_command,#space_command
210 ld hso_time,txd_time
211 br hso_isr_exit
212
213 send_mark:
214 ldb hso_command,#mark_command
215 ld hso_time,txd_time
216
217 hso_isr_exit:
218 popf
219 ret
220 $eject
221
222 init_receive:
223 ; Called to prepare the serial input process to find the leading edge of
224 ; a start bit.
225
226 ldb ioc0,#00000000b ; disconnect change detector
227 ldb hsi_mode,#00100000b ; negative edges on HSI 2
228
229 flush_fifo:
230 orb ios1_save,ios1
231 bbc ios1_save,7,flush_fifo_done
232 ldb al,hsi_status
233 ld ax,hsi_time ; trash the fifo entry

```

270061-94

```

0088 717F00      R    231      andb    ios1_save,#not(80h)      ; clear bit 7.
0088 27EF        232      br        flush_fifo
008D             233      flush_fifo_done:
008D B11015      234      ldb     ioc0,#00010000b      ; connect HSI.2 to detector
0090 F0         235      ret
0091             236
0091             237
0091             238
0091             239      hsi_isr:
240      ; Fields interrupts from the HSI unit, used to detect the leading edge
241      ; of the START bit
242      ; Note: this routine would be incorporated into the HSI strategy of an actual
243      ; system.
244      ;
2004             245      cseg at 2004h
2004 9100      R    246      dcw     hsi_isr      ; setup the interrupt vector
0091             247
0091             248      cseg
0091 F2         249      pushf
0092 C81C      250      push     ax
0094 B0061C      251      ldb     al,hsi_status
0097 A00404      R    252      ld      sample_time,hsi_time
009A 341C15      253      bbc     al,4,exit_hsi
009D 3F15FD      254      bbs     ios0,7,$      ; wait for room in HSO holding reg
00A0 A0081C      R    255      ld      ax,baud_count      ; send out sample command in 1/2
00A3 08011C      256      shr     ax,#1      ; bit time
00A6 641C04      R    257      add     sample_time,ax
00A9 B11806      258      ldb     hso_command,#sample_command
00AC C00404      R    259      st      sample_time,hso_time
00AF B10015      260      ldb     ioc0,#00000000b      ; disconnect hsi.2 from change detector
00B2             261      exit_hsi:
00B2 CC1C      262      pop     ax
00B4 F3        263      popf
00B5 F0        264      ret
00B5             265      $eject
00B5             266
00B6             267      software_timer_isr:
268      ; Fields the software timer interrupt, used to deserialize the incoming data.
269      ; Note: this routine would be incorporated into the software timer strategy
270      ; in an actual system.
271      ;
200A             272      cseg at 200ah
200A B600      R    273      dcw     software_timer_isr      ; setup vector
00B6             274
00B6             275      cseg
00B6 F2        276      pushf
00B7 901600      R    277      orb     ios1_save,ios1
00BA 71FE00      R    278      andb    ios1_save,#not(01h)      ; clear bit 0
00BD 51FC0100    R    279      andb    0,rcve_state,#0fch      ; All bits except rxrdy and overrun=0
00C1 D70C      280      bne     process_data

```

```

00C3          281  process_start_bit:
00C3 350604    282      bbc     hsi_status,5,start_ok
00C6 2FAE      283      call    init_receive
00C8 2032      284      br      software_timer_exit
00CA          285  start_ok:
00CA 910401    R 286      orb     rcve_state,#rip ; set receive in progress flag
00CD 2021      287      br      schedule_sample
                288
00CF          289  process_data:
00CF 3F010E    R 290      bbs     rcve_state,7,check_stopbit
00D2 180103    R 291      shrb    rcve_reg,#1
00D5 350603    292      bbc     hsi_status,5,datazero
00D8 918003    R 293      orb     rcve_reg,#80h ; set the new data bit
00DB          294  datazero:
00DB 751001    R 295      addb    rcve_state,#10h ; increment bit count
00DE 2010      296      br      schedule_sample
                297
00E0          298  check_stopbit:
00E0 3506FD    299      bbc     hsi_status,5,$ ; DEBUG ONLY
00E3 800302    R 300      ldb     rcve_buf,rcve_reg
00E6 910101    R 301      orb     rcve_state,#rxrdy
00E9 710301    R 302      andb    rcve_state,#03h ; Clear all but ready and overrun bits
00EC 2F88      303      call    init_receive
00EE 200C      304      br      software_timer_exit
                305
00F0          306  schedule_sample:
00F0 3F15FD    307      bbs     ios0,7,$ ; wait for holding reg empty
00F3 811806    R 308      ldb     hso_command,#sample_command
00F6 640804    R 309      add     sample_time,baud_count
00F9 C00404    R 310      st      sample_time,hso_time
                311
00FC          312  software_timer_exit:
00FC F3        313      popf
00FD F0        314      ret
                315
00FE          316  INT21H A1000
                317  end

```

ASSEMBLY COMPLETED. NO ERROR(S) FOUND.

270061-96

SERIES-III MCS-96 MACRO ASSEMBLER, V1 0

SOURCE FILE: F3.MOTCON.A96

OBJECT FILE: F3.MOTCON.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: NOSB

ERR LOC	OBJECT	LINE	SOURCE STATEMENT
		1	\$TITLE ('MOTCON.A96: Motor Control Example Program')
		2	
		3	USE WITH C-STEP or later parts
		4	
		5	December 20, 1984
		6	
		7	\$INCLUDE(DEMO96.INC)
		8	\$nolist ; Turn listing off for include file
		9	; End of include file
		10	
		11	Initial Values
		12	
	001E0	13	min_hsil_t equ 30 ; min period for PHA edges in model before mode2
	001C0	14	
	003C	15	min_hsil_t equ 2*min_hsil_t
		16	; min period for PHA edges in mode0 before model
	0069	17	max_hsil_t equ 3*min_hsil_t + min_hsil_t/2
		18	; max period for PHA edges in model before mode0
	006E	19	HS00_dly_period equ 110 ; delay for HSD timer 0 (timed count of pulses)
		20	; min period for 5 T2 clocks before mode 1
	00FA	21	swt1_dly_period equ 250 ; delay for software timer 1
	00FA	22	swt2_dly_period equ 250 ; delay for software timer 2
	00FF	23	max_power equ 0FFh
	00FF	24	max_brake equ 0FFh
	0080	25	maximum_hold equ 080H
	0480	26	brake_pnt equ 1200
	0064	27	position_pnt equ 100
	0010	28	velocity_pnt equ 16
		29	
	0024	30	RSEG at 024H
		31	
	0024	32	tmp: dsl 1
	0028	33	timer_2: dsl 1
		34	

270061-97

APPENDIX D MOTOR CONTROL PROGRAM

```

002C      86      tmr2_old:      dsl 1
0030      87      position:     dsl 1
0034 814510      88      des_pos:      dsl 1
0038 814505      89      pos_err:     dsl 1
003C      90      delta_p:      dsl 1
0040 0519      91      time:       dsl 1
0044 80002C      92      des_time:    dsl 1
0048 10494C      93      time_err:   dsl 1
0050 022C      94      $EJECT      15
0054 410132C    95      last_time_err dsw 1
0058 142C      96      last_pos_err: dsw 1
004C      97      pos_delta: dsw 1
004E 814513      98      time_delta: dsw 1
0050 4100010    99      last_pos: dsw 1
0052      100      last1_time: dsw 1
0054      101      last2_time: dsw 1
0056      102      boost:      dsw 1
0058      103      tmp1:       dsw 1
005A      104      out_ptr:    dsw 1
005C      105      offset:     dsw 1
005E      106      nxt_pos:    dsw 1
0060      107      rpwr:       dsw 1
0062 1050      108      old_t2:      dsw 1
0064 1030      109      direct:     dsw 1 ; 1=forward, 0=reverse
0066 5055      110      pwm_dir:    dsw 1
0068 8055      111      hsi_s0:      dsw 1
0069 0454      112      last_stat: dsw 1
006A 1050      113      pwm_pwr:     dsw 1
006B 0655      114      ios1_bak:    dsw 1
006C      115      TR_COL:   DSB 1 COLLECT TRACE IF TR_COL=00
006D      116      main_dly: dsw 1
006E      117      max_pwr:   dsw 1
006F      118      max_brk:   dsw 1
0070      119      max_hold: dsw 1
0072      120      vel_pnt:   dsw 1
0074      121      brk_pnt:   dsw 1
0076      122      pos_pnt:   dsw 1
0078      123      HS00_dly: dsw 1
007A      124      swt1_dly: dsw 1
007C      125      swt2_dly: dsw 1
007E      126      min_hsi:   dsw 1
0080      127      min_hsi1: dsw 1
0082      128      max_hsi1: dsw 1
0084      129      dseg_at: 100H
0086      130      dseg_at: 100H
0104      131      dseg_at: 100H
01005      132      dseg_at: 100H
0100      133      dseg_at: 100H

```



```

209D B12516      186      ldb      IOC1,#00100101B ; Disable HSD.4,HSD.5, HSI_INT=first,
209E B12517      187      andb     #00000001B ; Enable PWM,TXD,TIMER1_OVRFLOW_INT
209F B12518      188      lpc      #00000001B ;
20A0 71FC0F      189      andb     Port1,#11111100B ; clear P1.0.1 (set mode 0)
20A3 B19903      190      ldb      HSI_mode,#10011001B ; set hsi.1.3 -; hsi.0.2 +
20A6 B15715      191      ldb      IOC0,#01010111B ; Enable all hsi
20A7 B15716      192      lpc      #00100101B ; T2 CLOCK=T2CLK, T2RST=T2RST
20A8 B15717      193      lpc      #00100101B ; Clear timer2
20A9 B01400      194      *eject      old      #00100101B
20AA B01401      195      lpc      #00100101B
20AB B01402      196      lpc      #00100101B
20AC 0140        197      lpc      #00100101B
20AD 0140        198      lpc      #00100101B
20AE 0142        199      lpc      #00100101B
20AF 012B        200      lpc      #00100101B
20B0 012A        201      lpc      #00100101B
20B1 0130        202      lpc      #00100101B
20B2 0132        203      lpc      #00100101B
20B3 0134        204      lpc      #00100101B
20B4 0136        205      lpc      #00100101B
20B5 0138        206      lpc      #00100101B
20B6 0144        207      lpc      #00100101B
20B7 0146        208      lpc      #00100101B
20B8 A00A56      209      sub     last2_time,last1_time,#800H
20B9 4900085658  210      clrb     ios1_bak,#00100101B
20BA 116D        211      clrb     int_pending
20BB 1109        212      ld      out_ptr,#1FOH
20BC A1F0015E    213      ld      min_hsi,#min_hsi_t
20BD A13C0082    214      ld      min_hsil,#min_hsil_t
20BE A11E0084    215      ld      max_hsil,#max_hsil_t
20BF A1690086    216      ld      HSD0_dly,#HSD0_dly_period
20C0 A16E007C    217      ld      swt1_dly,#swt1_dly_period
20C1 A1FA007E    218      ld      swt2_dly,#(swt2_dly_period)
20C2 A1FA0080    219      ld      max_pwr,#max_power
20C3 A1FF0070    220      ld      max_brk,#max_brake
20C4 A1FF0072    221      ld      max_hold,#maximum_hold
20C5 A1800074    222      ld      brk_pnt,#brake_pnt
20C6 A1800478    223      ld      pos_pnt,#position_pnt
20C7 A164007A    224      ld      vel_pnt,#velocity_pnt
20C8 A1100076    225      ld      nxt_pos,#pos_table
20C9 A1002962    226      ldb      pwm_pwr,zero
20CA B0006C      227      ldb      pwm_dir,#01h ; FORWARD
20CB B10169      228      lpc      #00100101B
20CC B12D08      229      ldb      int_mask,#00101101B ; Enable tmr_ovf, hsi, swt, HSD.interrupts
20CD B13006      230      ldb      hso_command,#30H ; set HSD_O
20CE 447C0A04    231      add     hso_time,timer1,HSD0_dly
20CF 116 FD      232      nop
20D0 117 FD      233      NOP
20D1 B13906      234      ldb      hso_command,#39H ; set swt_1
20D2 447E0A04    235      add     hso_time,timer1,swt1_dly

```



```

211F FD          236      nop
2120 FD          237      nop
2121 B13A06      238      ldb  con:hso_command,#3AH ; set swt_2
2124 44800A04    239      add   hso_time.timer1,swt2_dly
2125 FD          240      nop
2128 A00A40      241      lds   time,TIMER1
212B A00C2C      242      lds   tmr2_old,timer2
212E FB          243      eld   hso_time.timer1,swt2_dly
212F E7CE06      244      nop
2130 B0000C      245      brn   all_main_prog
2133 00000000     246      nop
2136 00000000     247      $reject
2139 00000000     248      nop
213C 00000000     249      nop
213F 00000000     250      nop
2142 00000000     251      nop
2145 00000000     252      nop
2148 00000000     253      nop
214B 00000000     254      nop
214E 00000000     255      nop
2151 00000000     256      timer_ovf_int
2154 00000000     257      pushf
2157 00000000     258      nop
215A 00000000     259      nop
215D 00000000     260      chk_t1: jbc  hso_ios1_bak,IOS1
2160 00000000     261      inc  hso_time*2
2163 00000000     262      andb  hso_ios1_bak,#11011111B ; clear bit 5
2166 00000000     263      tmr_int_done
2169 00000000     264      popf
216C 00000000     265      ret ; End of timer interrupt routine
216F 00000000     266      nop
2172 00000000     267      nop
2175 00000000     268      nop
2178 00000000     269      nop
217B 00000000     270      nop
217E 00000000     271      nop
2181 00000000     272      nop
2184 00000000     273      nop
2187 00000000     274      nop
218A 00000000     275      nop
218D 00000000     276      soft_tmr_int
2190 00000000     277      pushf
2193 00000000     278      orb   ios1_bak,IOS1
2196 00000000     279      chk_sw0:
2199 00000000     280      jbc   ios1_bak,0.chk_sw0
219C 00000000     281      andb  hso_ios1_bak,#11111110B ; Clear bit 0 - end swt0
219F 00000000     282      call  hso_sw0_expired
21A2 00000000     283      chk_sw1:
21A5 00000000     284      jbc   ios1_bak,1.chk_sw1
21A8 00000000     285      andb  ios1_bak,#11111101B ; Clear bit 1
21AB 00000000     286      nop

```

270061-A1

270061-A2

```

22B3 643C30      336  in_fwd: add    position,delta_p
22B6 A40032      337      addc   position+2,zero
22B9 2006        338      br     chk_mode
22B8 683C30      339
22BE A80032      340  in_rev: sub    position,delta_p
22C1            341      subc   position+2,zero
22C1            342
22C1            343  chk_mode:
22C1 4866285C     344      sub     tmp1,Timer_2,old_t2      ; Check count difference in tmp1
22C5 8905005C     345      cmp     tmp1,#5                  ; set model if count is too low
22C9 D21C         346      jgt     end_swt0              ; count <= 5 HSI
22CB            347
22CB            348  set_model:
22CB 71FDOF       349      andb    Port1,#11111101B      ; Clear P1.1, set P1.0 (set mode 1)
22CF 91010F       350      orb     Port1,#00000001B
22D1 B10515       351      ldb     IDCO,#00000101B          ; enable HSI 0 and 1
22D4 A00400       352      ld      zero,HSI_TIME
22D7 48B40A56     353      sub     last1_time,Timer1,min_hsi1
22D7            354      ; set up so (time-last2_time)>min_hsi1 on next HSI
22D7            355  $EJECT
22D7            356
22DB            357  clr_hsi:
22DB A00400       358      ld      ZERO,HSI_TIME
22DE 717F6D       359      andb    ios1_bak,#01111111B      ; clear bit 7
22E1 90166D       360      orb     ios1_bak,ios1
22E4 3F6DF4       361      jbs     ios1_bak,7,clr_hsi      ; If hsi is triggered then clear hsi
22E4            362
22E7            363  end_swt0:
22E7 A02866       364      ld      old_t2,TIMER_2
22EA 71DF0F       365      andb    port1,#11011111B      ; clear P1.5
22ED F3           366      POPF
22EE F0           367      ret
22EE            368
22EE            369
22EE            370
22EE            371
22EE            372  SOFTWARE TIMER ROUTINE 2
22EE            373
22EE            374
22EE            375  CSEQ AT 2380H
22EE            376
22EE            377  swt2_expired:
22EE            378      pushf
22EE            379      ldb     hso_command,#3AH          ; set swt_2
22EE            380      add     hso_time,Timer1,swt2_dly
22EE            381
22EE            382      orb     port1,#00000100B          ; set port 1.2
22EE            383      cmp     out_ptr,#7ffH
22EE            384      bnh     pulsing
22EE            385      ld      out_ptr,#1f0H

```

```

2395 0105          386
2395 306E0C        387 pulsing: 18
2395 306E0C        388 jbc     tr_col,0,swt2_done
2395 306E0C        389
239B C25F32        390 st      position+2,[out_ptr]+ ; position high, position low
239B C25F30        391 st      position,[out_ptr]+
239E C25F68        392
239A C25F6C        393 st      direct,[out_ptr]+
239A C25F6C        394 st      pwm_pwm,[out_ptr]+
239A C25F6C        395
239A C25F6C        396
239A C25F6C        397
239A C25F6C        398
239A C25F6C        399 swt2_done:
239A C25F6C        400 sub     tmp1,timer1,last1_time
239A C25F6C        401 cmp     tmp1,#1800H
239A C25F6C        402 jnh     swt2_ret ; keep (timer1-last1_time)<2000H
239A C25F6C        403 add     last1_time,#1000H
239A C25F6C        404
239A C25F6C        405 swt2_ret:
239A C25F6C        406 andb   port1,#11111011B ; clear port1.2
239A C25F6C        407 popf
239A C25F6C        408 ret
239A C25F6C        409
239A C25F6C        410 $EJECT
239A C25F6C        411
239A C25F6C        412
239A C25F6C        413
239A C25F6C        414 ; This routine keeps track of the current time and position of the motor.
239A C25F6C        415 ; The upper word of information is provided by the timer overflow routine.
239A C25F6C        416
239A C25F6C        417 CSEG AT 2400H
239A C25F6C        418 now_mode_1: br     in_mode_1 ; used to save execution time for
239A C25F6C        419 no_int1: br     no_int ; worst case loop
239A C25F6C        420
239A C25F6C        421 hsi_data_int: pushf
239A C25F6C        422 orb     port1,#01000000B ; set P1.6
239A C25F6C        423 andb    ios1_bak,#01111111B ; Clear ios1_bak.7
239A C25F6C        424 orb     ios1_bak,ios1
239A C25F6C        425 jbc     ios1_bak,7,no_int1 ; If hsi is not triggered then
239A C25F6C        426 ; jump to no_int
239A C25F6C        427 get_values:
239A C25F6C        428 ld      timer_2,TIMER2
239A C25F6C        429 andb    hsi_s0,HSI_STATUS,#01010101B
239A C25F6C        430 ld      time,HSI_TIME
239A C25F6C        431
239A C25F6C        432 jbs     port1,0,now_mode_1 ; jump if in mode 1
239A C25F6C        433
239A C25F6C        434 In_mode_0:
239A C25F6C        435 jbs     hsi_s0,0,a_rise

```

270061-A4


```

2421 3A6A2C      436      jbs      hsi_s0.2,a_fall
2424 3C6A4D      437      jbs      hsi_s0.4,b_rise
2427 3E6A5A      438      jbs      hsi_s0.6,b_fall
242A 2094        439      br       no_cnt
242C A05658      440
242F A04056      441      a_rise: ld      last2_time,last1_time
2432 685840      442      ld      last1_time,time
2435 888240      443      sub      time,last2_time
2438 D906        444      cmp      time,min_hsi
243A 91010F      445      jh       tst_statf
243D B10515      446      ;set model-
2440            447      orb      Port1,#00000001B      ; Set P1.0 (in mode 1)
2443 3E6B5B      448      ldb      IOCO,#00000101B      ; Enable HSI 0 and 1
2446 3A6B50      449      tst_statf
2449 98006B      450      jbs      last_stat,6,going_fwd
244C DF46        451      jbs      last_stat,4,going_rev
244E 27B2        452      jbs      last_stat,2,change_dir
2450 A05658      453      cmpb     last_stat,zero
2453 A04056      454      je       first_time      ; first time in mode0
2456 685840      455      br       no_int1
2459 888240      456
245C D906        457      a_fall: ld      last2_time,last1_time
245F            458      ld      last1_time,time
2462            459      sub      time,last2_time
2465            460      cmp      time,min_hsi
2468            461      jh       tst_statf
246B            462      ;set model-
246E            463      orb      Port1,#00000001B      ; Set P1.0 (in mode 1)
2471            464      ldb      IOCO,#00000101B      ; Enable HSI 0 and 1
2474            465      $EJECT
2477            466      tst_statf
247A            467      jbs      last_stat,4,going_fwd
247D            468      jbs      last_stat,6,going_rev
2480            469      jbs      last_stat,0,change_dir
2483            470      cmpb     last_stat,zero
2486            471      je       first_time      ; first time in mode0
2489            472      br       no_int
248C            473
248F            474      b_rise: jbs      last_stat,0,going_fwd
2492            475      jbs      last_stat,2,going_rev
2495            476      jbs      last_stat,6,change_dir
2498            477      cmpb     last_stat,zero
249B            478      je       first_time      ; first time in mode0
249E            479      br       no_int
24A1            480
24A4            481      b_fall: jbs      last_stat,2,going_fwd
24A7            482      jbs      last_stat,0,going_rev
24AA            483      jbs      last_stat,4,change_dir
24AD            484      cmpb     last_stat,zero
24B0            485      je       first_time      ; first time in mode0
24B3            486
24B6            487
24B9            488
24BC            489
24BF            490
24C2            491
24C5            492
24C8            493
24CB            494
24CE            495
24D1            496
24D4            497
24D7            498
24DA            499
24DD            500
24E0            501
24E3            502
24E6            503
24E9            504
24EC            505
24EF            506
24F2            507
24F5            508
24F8            509
24FB            510
24FE            511
2501            512
2504            513
2507            514
250A            515
250D            516
2510            517
2513            518
2516            519
2519            520
251C            521
251F            522
2522            523
2525            524
2528            525
252B            526
252E            527
2531            528
2534            529
2537            530
253A            531
253D            532
2540            533
2543            534
2546            535
2549            536
254C            537
254F            538
2552            539
2555            540
2558            541
255B            542
255E            543
2561            544
2564            545
2567            546
256A            547
256D            548
2570            549
2573            550
2576            551
2579            552
257C            553
257F            554
2582            555
2585            556
2588            557
258B            558
258E            559
2591            560
2594            561
2597            562
259A            563
259D            564
25A0            565
25A3            566
25A6            567
25A9            568
25AC            569
25AF            570
25B2            571
25B5            572
25B8            573
25BB            574
25BE            575
25C1            576
25C4            577
25C7            578
25CA            579
25CD            580
25D0            581
25D3            582
25D6            583
25D9            584
25DC            585
25DF            586
25E2            587
25E5            588
25E8            589
25EB            590
25EE            591
25F1            592
25F4            593
25F7            594
25FA            595
25FD            596
2600            597
2603            598
2606            599
2609            600
260C            601
260F            602
2612            603
2615            604
2618            605
261B            606
261E            607
2621            608
2624            609
2627            610
262A            611
262D            612
2630            613
2633            614
2636            615
2639            616
263C            617
263F            618
2642            619
2645            620
2648            621
264B            622
264E            623
2651            624
2654            625
2657            626
265A            627
265D            628
2660            629
2663            630
2666            631
2669            632
266C            633
266F            634
2672            635
2675            636
2678            637
267B            638
267E            639
2681            640
2684            641
2687            642
268A            643
268D            644
2690            645
2693            646
2696            647
2699            648
269C            649
269F            650
26A2            651
26A5            652
26A8            653
26AB            654
26AE            655
26B1            656
26B4            657
26B7            658
26BA            659
26BD            660
26C0            661
26C3            662
26C6            663
26C9            664
26CC            665
26CF            666
26D2            667
26D5            668
26D8            669
26DB            670
26DE            671
26E1            672
26E4            673
26E7            674
26EA            675
26ED            676
26F0            677
26F3            678
26F6            679
26F9            680
26FC            681
26FF            682
2700            683
2703            684
2706            685
2709            686
270C            687
270F            688
2712            689
2715            690
2718            691
271B            692
271E            693
2721            694
2724            695
2727            696
272A            697
272D            698
2730            699
2733            700
2736            701
2739            702
273C            703
273F            704
2742            705
2745            706
2748            707
274B            708
274E            709
2751            710
2754            711
2757            712
275A            713
275D            714
2760            715
2763            716
2766            717
2769            718
276C            719
276F            720
2772            721
2775            722
2778            723
277B            724
277E            725
2781            726
2784            727
2787            728
278A            729
278D            730
2790            731
2793            732
2796            733
2799            734
279C            735
279F            736
27A2            737
27A5            738
27A8            739
27AB            740
27AE            741
27B1            742
27B4            743
27B7            744
27BA            745
27BD            746
27C0            747
27C3            748
27C6            749
27C9            750
27CC            751
27CF            752
27D2            753
27D5            754
27D8            755
27DB            756
27DE            757
27E1            758
27E4            759
27E7            760
27EA            761
27ED            762
27F0            763
27F3            764
27F6            765
27F9            766
27FC            767
27FF            768
2800            769
2803            770
2806            771
2809            772
280C            773
280F            774
2812            775
2815            776
2818            777
281B            778
281E            779
2821            780
2824            781
2827            782
282A            783
282D            784
2830            785
2833            786
2836            787
2839            788
283C            789
283F            790
2842            791
2845            792
2848            793
284B            794
284E            795
2851            796
2854            797
2857            798
285A            799
285D            800
2860            801
2863            802
2866            803
2869            804
286C            805
286F            806
2872            807
2875            808
2878            809
287B            810
287E            811
2881            812
2884            813
2887            814
288A            815
288D            816
2890            817
2893            818
2896            819
2899            820
289C            821
289F            822
28A2            823
28A5            824
28A8            825
28AB            826
28AE            827
28B1            828
28B4            829
28B7            830
28BA            831
28BD            832
28C0            833
28C3            834
28C6            835
28C9            836
28CC            837
28CF            838
28D2            839
28D5            840
28D8            841
28DB            842
28DE            843
28E1            844
28E4            845
28E7            846
28EA            847
28ED            848
28F0            849
28F3            850
28F6            851
28F9            852
28FC            853
28FF            854
2900            855
2903            856
2906            857
2909            858
290C            859
290F            860
2912            861
2915            862
2918            863
291B            864
291E            865
2921            866
2924            867
2927            868
292A            869
292D            870
2930            871
2933            872
2936            873
2939            874
293C            875
293F            876
2942            877
2945            878
2948            879
294B            880
294E            881
2951            882
2954            883
2957            884
295A            885
295D            886
2960            887
2963            888
2966            889
2969            890
296C            891
296F            892
2972            893
2975            894
2978            895
297B            896
297E            897
2981            898
2984            899
2987            900
298A            901
298D            902
2990            903
2993            904
2996            905
2999            906
299C            907
299F            908
29A2            909
29A5            910
29A8            911
29AB            912
29AE            913
29B1            914
29B4            915
29B7            916
29BA            917
29BD            918
29C0            919
29C3            920
29C6            921
29C9            922
29CC            923
29CF            924
29D2            925
29D5            926
29D8            927
29DB            928
29DE            929
29E1            930
29E4            931
29E7            932
29EA            933
29ED            934
29F0            935
29F3            936
29F6            937
29F9            938
29FC            939
29FF            940
2A00            941
2A03            942
2A06            943
2A09            944
2A0C            945
2A0F            946
2A12            947
2A15            948
2A18            949
2A1B            950
2A1E            951
2A21            952
2A24            953
2A27            954
2A2A            955
2A2D            956
2A30            957
2A33            958
2A36            959
2A39            960
2A3C            961
2A3F            962
2A42            963
2A45            964
2A48            965
2A4B            966
2A4E            967
2A51            968
2A54            969
2A57            970
2A5A            971
2A5D            972
2A60            973
2A63            974
2A66            975
2A69            976
2A6C            977
2A6F            978
2A72            979
2A75            980
2A78            981
2A7B            982
2A7E            983
2A81            984
2A84            985
2A87            986
2A8A            987
2A8D            988
2A90            989
2A93            990
2A96            991
2A99            992
2A9C            993
2A9F            994
2AA2            995
2AA5            996
2AA8            997
2AAB            998
2AAE            999
2AB1            1000
2AB4            1001
2AB7            1002
2ABA            1003
2ABD            1004
2AC0            1005
2AC3            1006
2AC6            1007
2AC9            1008
2ACC            1009
2ACF            1010
2AD2            1011
2AD5            1012
2AD8            1013
2ADB            1014
2ADE            1015
2AE1            1016
2AE4            1017
2AE7            1018
2AEA            1019
2AED            1020
2AF0            1021
2AF3            1022
2AF6            1023
2AF9            1024
2AFC            1025
2AFF            1026
2B00            1027
2B03            1028
2B06            1029
2B09            1030
2B0C            1031
2B0F            1032
2B12            1033
2B15            1034
2B18            1035
2B1B            1036
2B1E            1037
2B21            1038
2B24            1039
2B27            1040
2B2A            1041
2B2D            1042
2B30            1043
2B33            1044
2B36            1045
2B39            1046
2B3C            1047
2B3F            1048
2B42            1049
2B45            1050
2B48            1051
2B4B            1052
2B4E            1053
2B51            1054
2B54            1055
2B57            1056
2B5A            1057
2B5D            1058
2B60            1059
2B63            1060
2B66            1061
2B69            1062
2B6C            1063
2B6F            1064
2B72            1065
2B75            1066
2B78            1067
2B7B            1068
2B7E            1069
2B81            1070
2B84            1071
2B87            1072
2B8A            1073
2B8D            1074
2B90            1075
2B93            1076
2B96            1077
2B99            1078
2B9C            1079
2B9F            1080
2BA2            1081
2BA5            1082
2BA8            1083
2BAB            1084
2BAE            1085
2BB1            1086
2BB4            1087
2BB7            1088
2BBA            1089
2BBD            1090
2BC0            1091
2BC3            1092
2BC6            1093
2BC9            1094
2BCC            1095
2BCF            1096
2BD2            1097
2BD5            1098
2BD8            1099
2BDB            1100
2BDE            1101
2BE1            1102
2BE4            1103
2BE7            1104
2BEA            1105
2BED            1106
2BF0            1107
2BF3            1108
2BF6            1109
2BF9            1110
2BFC            1111
2BFF            1112
2C00            1113
2C03            1114
2C06            1115
2C09            1116
2C0C            1117
2C0F            1118
2C12            1119
2C15            1120
2C18            1121
2C1B            1122
2C1E            1123
2C21            1124
2C24            1125
2C27            1126
2C2A            1127
2C2D            1128
2C30            1129
2C33            1130
2C36            1131
2C39            1132
2C3C            1133
2C3F            1134
2C42            1135
2C45            1136
2C48            1137
2C4B            1138
2C4E            1139
2C51            1140
2C54            1141
2C57            1142
2C5A            1143
2C5D            1144
2C60            1145
2C63            1146
2C66            1147
2C69            1148
2C6C            1149
2C6F            1150
2C72            1151
2C75            1152
2C78            1153
2C7B            1154
2C7E            1155
2C81            1156
2C84            1157
2C87            1158
2C8A            1159
2C8D            1160
2C90            1161
2C93            1162
2C96            1163
2C99            1164
2C9C            1165
2C9F            1166
2CA2            1167
2CA5            1168
2CA8            1169
2CAB            1170
2CAE            1171
2CB1            1172
2CB4            1173
2CB7            1174
2CBA            1175
2CBD            1176
2CC0            1177
2CC3            1178
2CC6            1179
2CC9            1180
2CCC            1181
2CCF            1182
2CD2            1183
2CD5            1184
2CD8            1185
2CDB            1186
2CDE            1187
2CE1            1188
2CE4            1189
2CE7            1190
2CEA            1191
2CED            1192
2CF0            1193
2CF3            1194
2CF6            1195
2CF9            1196
2CFC            1197
2CFF            1198
2D00            1199
2D03            1200
2D06            1201
2D09            1202
2D0C            1203
2D0F            1204
2D12            1205
2D15            1206
2D18            1207
2D1B            1208
2D1E            1209
2D21            1210
2D24            1211
2D27            1212
2D2A            1213
2D2D            1214
2D30            1215
2D33            1216
2D36            1217
2D39            1218
2D3C            1219
2D3F            1220
2D42            1221
2D45            1222
2D48            1223
2D4B            1224
2D4E            1225
2D51            1226
2D54            1227
2D57            1228
2D5A            1229
2D5D            1230
2D60            1231
2D63            1232
2D66            1233
2D69            1234
2D6C            1235
2D6F            1236
2D72            1237
2D75            1238
2D78            1239
2D7B            1240
2D7E            1241
2D81            1242
2D84            1243
2D87            1244
2D8A            1245
2D8D            1246
2D90            1247
2D93            1248
2D96            1249
2D99            1250
2D9C            1251
2D9F            1252
2DA2            1253
2DA5            1254
2DA8            1255
2DAB            1256
2DAE            1257
2DB1            1258
2DB4            1259
2DB7            1260
2DBA            1261
2DBD            1262
2DC0            1263
2DC3            1264
2DC6            1265
2DC9            1266
2DCC            1267
2DCF            1268
2DD2            1269
2DD5            1270
2DD8            1271
2ddb            1272
2DDE            1273
2DE1            1274
2DE4            1275
2DE7            1276
2DEA            1277
2DED            1278
2DF0            1279
2DF3            1280
2DF6            1281
2DF9            1282
2DFC            1283
2DFF            1284
2E00            1285
2E03            1286
2E06            1287
2E09            1288
2E0C            1289
2E0F            1290
2E12            1291
2E15            1292
2E18            1293
2E1B            1294
2E1E            1295
2E21            1296
2E24            1297
2E27            1298
2E2A            1299
2E2D            1300
2E30            1301
2E33            1302
2E36            1303
2E39            1304
2E3C            1305
2E3F            1306
2E42            1307
2E45            1308
2E48            1309
2E4B            1310
2E4E            1311
2E51            1312
2E54            1313
2E57            1314
2E5A            1315
2E5D            1316
2E60            1317
2E63            1318
2E66            1319
2E69            1320
2E6C            1321
2E6F            1322
2E72            1323
2E75            1324
2E78            1325
2E7B            1326
2E7E            1327
2E81            1328
2E84            1329
2E87            1330
2E8A            1331
2E8D            1332
2E90            1333
2E93            1334
2E96            1335
2E99            1336
2E9C            1337
2E9F            1338
2EA2            1339
2EA5            1340
2EA8            1341
2EAB            1342
2EAE            1343
2EB1            1344
2EB4            1345
2EB7            1346
2EBA            1347
2EBD            1348
2EC0            1349
2EC3            1350
2EC6            1351
2EC9            1352
2ECC            1353
2ECF            1354
2ED2            1355
2ED5            1356
2ED8            1357
2EDB            1358
2EDE            1359
2EE1            1360
2EE4            1361
2EE7            1362
2EEA            1363
2EED            1364
2EF0            1365
2EF3            1366
2EF6            1367
2EF9            1368
2EFC            1369
2EFF            1370
2F00            1371
2F03            1372
2F06            1373
2F09            1374
2F0C            1375
2F0F            1376
2F12            1377
2F15            1378
2F18            1379
2F1B            1380
2F1E            1381
2F21            1382
2F24            1383
2F27            1384
2F2A            1385
2F2D            1386
2F30            1387
2F33            1388
2F36            1389
2F39            1390
2F3C            1391
2F3F            1392
2F42            1393
2F45            1394
2F48            1395
2F4B            1396
2F4E            1397
2F51            1398
2F54            1399
2F57            1400
2F5A            1401
2F5D            1402
2F60            1403
2F63            1404
2F66            1405
2F69            1406
2F6C            1407
2F6F            1408
2F72            1409
2F75            1410
2F78            1411
2F7B            1412
2F7E            1413
2F81            1414
2F84            1415
2F87            1416
2F8A            1417
2F8D            1418
2F90            1419
2F93            1420
2F96            1421
2F99            1422
2F9C            1423
2F9F            1424
2FA2            1425
2FA5            1426
2FA8            1427
2FAB            1428
2FAE            1429
2FB1            1430
2FB4            1431
2FB7            1432
2FBA            1433
2FBD            1434
2FC0            1435
2FC3            1436
2FC6            1437
2FC9            1438
2FCC            1439
2FCF            1440
2FD2            1441
2FD5            1442
2FD8            1443
2FDB            1444
2FDE            1445
2FE1            1446
2FE4            1447
2FE7            1448
2FEA            1449
2FED            1450
2FF0            1451
2FF3            1452
2FF6            1453
2FF9            1454
2FFC            1455
2FFF            1456
2G00            1457
2G03            1458
2G06            1459
2G09            1460
2G0C            1461
2G0F            1462
2G12            1463
2G15            1464
2G18            1465
2G1B            1466
2G1E            1467
2G21            1468
2G24            1469
2G27            1470
2G2A            1471
2G2D            1472
2G30            1473
2G33            1474
2G36            1475
2G39            1476
2G3C            1477
2G3F            1478
2G42            1479
2G45            1480
2G48            1481
2G4B            1482
2G4E            1483
2G51            1484
2G54            1485
2G57            1486
2G5A            1487
2G5D            1488
2G60            1489
2G63            1490
2G66            1491
2G69            1492
2G6C            1493
2G6F            1494
2G72            1495
2G75            1496
2G78            1497
2G7B            1498
2G7E            1499
2G81            1500
2G84            1501
2G87            1502
2G8A            1503
2G8D            1504
2G90            1505
2G93            1506
2G96            1507
2G99            1508
2G9C            1509
2G9F            1510
2GA2            1511
2GA5            1512
2GA8            1513
2GAB            1514
2GAE            1515
2GB1            1516
2GB4            1517
2GB7            1518
2GBA            1519
2GBD            1520
2GC0            1521
2GC3            1522
2GC6            1523
2GC9            1524
2GCC            1525
2GCF            1526
2GD2            1527
2GD5            1528
2GD8            1529
2GDB            1530
2GDE            1531
2GE1            1532
2GE4            1533
2GE7            1534
2GEA            1535
2GED            1536
2GF0            1537
2GF3            1538
2GF6            1539
2GF9            1540
2GFC            1541
2GFF            1542
2H00            1543
2H03            1544
2H06            1545
2H09            1546
2H0C            1547
2H0F            1548
2H12            1549
2H15            1550
2H18            1551
2H1B            1552
2H1E            1553
2H21            1554
2H24            1555
2H27            1556
2H2A            1557
2H2D            1558
2H30            1559
2H33            1560
2H36            1561
2H39            1562
2H3C            1563
2H3F            1564
2H42            1565
2H45            1566
2H48            1567
2H4B            1568
2H4E            1569
2H51            1570
2H54            1571
2H57            1572
2H5A            1573
2H5D            1574

```

```

2492 2037          486          br          no_int
2493 2038          487
2494 2039          488 first_time: 488
2495 2040          489          stb          hsi_s0,last_stat
2496 2041          490          bra          done_chk
2497 2042          491
2498 2043          492
2499 2044          493 change_dir: 493
2500 2045          494          notb         direct
2501 2046          495 no_inc: jbc          direct,0,going_rev
2502 2047          496
2503 2048          497 going_fwd: 497
2504 2049          498          orb          PORT2,#01000000B ; set P2.6
2505 2050          499          ldb          direct,#01 ; direction = forward
2506 2051          500          add          position,#01
2507 2052          501          addc         position+2,zero
2508 2053          502          bra          st_stat
2509 2054          503 going_rev: 503
2510 2055          504          andb         PORT2,#10111111B ; clear P2.6
2511 2056          505          ldb          direct,#00 ; direction = reverse
2512 2057          506          sub          position,#01
2513 2058          507          subc         position+2,zero
2514 2059          508
2515 2060          509 st_stat: 509
2516 2061          510          stb          hsi_s0,last_stat
2517 2062          511 load_last: 511
2518 2063          512          ld          tmr2_old,timer_2
2519 2064          513 no_cnt: andb         ios1_bak,#01111111B ; clr bit 7
2520 2065          514          orb          ios1_bak,ios1
2521 2066          515          jbc          ios1_bak,7,no_int
2522 2067          516          again: br          get_values
2523 2068          517
2524 2069          518 no_int: andb         port1,#10111111B ; Clear P1.6
2525 2070          519          popf         ; end of hsi_data interrupt routine
2526 2071          520          ret          ; Routine for mode 1 follows and then returns to "load_last"
2527 2072          521
2528 2073          522 %EJECT
2529 2074          523
2530 2075          524
2531 2076          525 In_mode_1: 525
2532 2077          526
2533 2078          527          andb         tmp1,hsi_s0,#01010000B
2534 2079          528          jne          no_cnt
2535 2080          529 cmp_time: 529
2536 2081          530          cmp          last2_time,last1_time ; Procedure which sets mode 1 also
2537 2082          531          ld          last1_time,time ; sets times to pass the tests
2538 2083          532          ld          last1_time,time
2539 2084          533
2540 2085          534 cmp1: 534
2541 2086          535          sub          tmp1,time,last2_time
2542 2087          536          cmp          tmp1,min_hsil
2543 2088          537

```

```

24E3 D914          536      jh      check_max_time
24E5 4B8B400C      537      sub     $0,0,$0
24E5 4B8B400C      538      set_mode_2:
24E5 91020F        539      orb     Port1,#00000010B      ; Set P1.1 (in mode 2)
24E8 B10015        540      ldb     IOC0,#00000000B      ; Disable all HSI
24E8 A00400        541      mt_hsi: ld     zero,hsi_time      ; empty the hsi fifo
24EE 717F6D        542      andb    ios1_bak,#01111111B      ; clear bit 7
24F1 90166D        543      orb     ios1_bak,ios1      ; clear bit 7
24F4 3F6DF4        544      jbs     ios1_bak,7,mt_hsi      ; If hsi is triggered then clear hsi
24F7 201273C      545      breq    done_chk,#0,$01010000B
24F9              546
24F9 4858405C      547      check_max_time:
24FD 88B65C        548      sub     tmp1,time,last2_time
24FD 88B65C        549      cmp     tmp1,max_hsi1      ; max_hsi = addition to min_hsi for
550              550      ; total time
551      jnh     done_chk      ; done_chk = 1 if max_hsi > min_hsi
552
553      set_mode_0:
554      andb    Port1,#11111100B      ; clear P1.0,1 set mode 00
555      ldb     IOC0,#01010101B      ; Enable all HSI
556      ldb     last_stat,zero
557
558      done_chk:
559      sub     delta_p,timer_2,tmp2_old      ; get timer2 count difference
560      jbc     direct,0,add_rev
561
562      add_fwd:
563      add     position,delta_p
564      addcc   position+2,zero
565      br      load_last
566
567      add_rev:
568      sub     position,delta_p
569      subcc   position+2,zero
570      br      load_last
571
572      $reject    PL      ; reject
573
574      CSEG:AT 2600H
575
576      swt1_expired:
577      ucb     $0,$0,$0
578      pushf
579      orb     port1,#10000000B      ; set port1.7
580
581      ldb     int_mask,#00001101B      ; enable HSI, Tovf, HSO
582
583      HSO_COMMAND,#39H
584
585      add     HSO_TIME,TIMER1,swt1_dly
586
587      PL      ; return

```

270061-A7

```

260E A0464A      586
2611 A0363A      587      ld      time_err+2,des_time+2      ; Calculate time & position error
2614 48404448     588      ld      pos_err+2,des_pos+2
2618 A8424A      589      sub     time_err,des_time,time      ; values are set
261B 48303438     590      subc    time_err+2,time+2
261F AB323A      591      sub     pos_err,des_pos,position
2622 FB          592      subc    pos_err+2,position+2
2623 48484C52     593
2627 A04B4C      594      EI
262A 483B4E50     595
262E A03B4E      596      sub     time_delta,last_time_err,time_err
2631 88003A      597      ld      last_time_err,time_err
2634 D60D        598
2636 033B        599      sub     pos_delta,last_pos_err,pos_err
2638 B10069      600      ld      last_pos_err,pos_err
263B 89FFFF3A     601
263F D70A        602      Time_err = Desired time to finish - current time
2641 200D        603      Pos_err = Desired position to finish - current position
2643 B10169      604      Pos_delta = Last position error - Current position error
2646 88003A      605      Time_delta = Last time error - Current time error
2649 DF05        606      note that errors should get smaller so deltas will be
264B B0706C      607      positive for forward motion (time is always forward)
264E 2051        608
2650 887A3B      609      chk_dir:
2653 D11E        610      cmp     pos_err+2,zero
2655 88783B      611      jge     go_forward
2658 88783B      612
265B 88783B      613      go_backward:
265E 88783B      614      neg     pos_err      ; Pos_err = ABS VAL (pos_err)
2661 88783B      615      ldb     pwm_dir,#00h
2664 88783B      616      cmp     pos_err+2,#0ffffH
2667 88783B      617      jne     ld_max
266A 88783B      618      br     chk_brk
266D 88783B      619
2670 88783B      620
2673 88783B      621      go_forward:
2676 88783B      622      ldb     pwm_dir,#01H
2679 88783B      623      cmp     pos_err+2,zero
267C 88783B      624      je      chk_brk
267F 88783B      625      $EJECT
2682 88783B      626
2685 88783B      627      ld_max: ldb     pwm_pwr,max_pwr
2688 88783B      628      br     chk_sanity
268B 88783B      629
268E 88783B      630      Chk_brk:
2691 88783B      631      cmp     pos_err,pos_pnt
2694 88783B      632      jnh     hold_position      ; position_error<position_control_point
2697 88783B      633      cmp     pos_err,brk_pnt

```

270061-A8


```

2658 D9F1      634      jh      ld_max      , position_error>brake_point
265A      635
265A      636      braking:
265A 880050      637      cmp      pos_delta,zero
265D D602      638      jge      chk_delta
265F 0350      639      neg      pos_delta
2661      640      chk_delta:
2661 887650      641      cmp      pos_delta,vel_pnt ; velocity = pos_delta/sample_time
2664 D10D      642      jnh      hold_position ; jmp if ABS(velocity) < vel_pnt
2666      643
2666 B0726C      644      brake: ldb      pwm_pwr,max_brk
2669 B06824      645      ldb      tmp,direct ; If braking apply power in opposite
266C 1224      646      notb     tmp ; direction of current motion
266E B02469      647      ldb      pwm_dir,tmp
2671 2030      648
2671      649      br      ld_pwr
2673      650
2673      651      Hold_position: ; position hold mode
2673 8902003B      652      cmp      pos_err,#02
2677 D906      653      jh      calc_out ; if position error < 2 then turn off power
2679 0126      654      clr      tmp+2
267B 015A      655      clr      boost
267D 201F      656      BR      output
267F      657
267F      658      calc_out:
267F 5DFF7424      659      mulub    tmp,max_hold,#255
2683 6C3824      660      mulu     tmp,pos_err ; Tmp = pos_err * max_hold
2686 880050      661      cmp      pos_delta,zero
2689 D709      662      jne      no_bst
268B 6504005A      663      add      boost,#04 ; Boost is integral control
268F 645A26      664      add      tmp+2,boost ; TMP+2 = MSB(pos_err*max_hold)
2692 2002      665      br      ck_max
2694 015A      666      no_bst: clr      boost
2696 887426      667      ck_max: cmp      tmp+2,max_hold
2699 D103      668      jnh      output
269B A07426      669      maxed: ld      tmp+2,max_hold
269E B0266C      670      output: ldb      pwm_pwr,tmp+2
26A1      671
26A1 2000      672
26A1      673      chk_sanity:
26A1      674      br      ld_pwr
26A3      675      ;;
26A3      676      ;;
26A3      677      $EJECT EI
26A3      678
26A3      679      ld_pwr:
26A3 B06C64      680      ldb      rpwr,pwm_pwr
26A6 1264      681      notb     rpwr
26A8 38690A      682      jbs      pwm_dir,0,p2fwd
26A8      683

```

270061-B0

```

280F          734 control:
280F 912D08    735      orb    int_mask, #00101101B      ; enable hsi, hso, swt, tovf interrupts
2812 FD       736      nop
2813 FD       737      nop
2814 FD       738      nop
2815 E06FFD    739      djnz   main_dly, $
2818 FD       740      nop
2819 95080F    741      xorb   port1, #00001000B      ; compliment p1.3
281C 27E2     742      BR      MAIN_PROG
              743
              744
2900          745 CSEQ AT 2900H
              746
2900          747 pos_table:
              748
2900 00000000  749      dcl    00000000H      ; position 0
2904 20008000  750      dcw    0020H, 0080H      ; next time, power
2908 00C00000  751      dcl    0000C000H      ; position 1
290C 40004000  752      dcw    0040H, 0040H      ; next time, power
2910 00000000  753      dcl    00000000H      ; position 2
2914 6000C000  754      dcw    0060H, 00C0H      ; next time, power
2918 0080FFFF  755      dcl    0FFFF8000H      ; position 3
291C 80008000  756      dcw    0080H, 0080H      ; next time, power
              757
2920 00080000  758      dcl    00000800H      ; position 4
2924 58008000  759      dcw    0058H, 0080H      ; next time, power
2928 00300000  760      dcl    00003000H      ; position 5
292C 7000FF00  761      dcw    0070H, 00FFH      ; next time, power
2930 00000000  762      dcl    00000000H      ; position 6
2934 9000F000  763      dcw    0090H, 00F0H      ; next time, power
2938 00000000  764      dcl    00000000H      ; position 7
293C 9100F000  765      dcw    0091H, 00F0H      ; next time, power
              766
              767
2940          768      END
ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

```



APPLICATION NOTE

AP-406

The light methods of processing analog signals. This Application Note assists with the first task—understanding of an analog acquisition system.

Designers experienced with analog design, or analog acquisition systems, may find no revelation herein. To those unfamiliar with analog acquisition systems, this Application Note provides a tutorial on the subject and will serve as a handy reference.

Answering the countless number of analog circuit design questions is beyond the scope of this Application Note. It is to say that the effort placed on the design of analog circuits should increase with a decreasing error budget.

At a minimum, the application literature of op-amp manufacturers and analog design manuals are a good place to start. Furthermore, the application literature of monolithic analog acquisition system manufacturers should be consulted since the suggestions presented therein are largely transparent to any A/D system.

This Application Note is organized in the following sections. The components of an analog acquisition system and the errors associated with each are first explained. Then, interfacing suggestions and ideas for getting more information are presented. Finally, a set of applications provides back-up information, a bibliography, actual converter data and some program listings.

For any user of an MCS-96 analog acquisition system (experienced or not), this document contains very useful information. It should be considered mandatory reading in addition to the latest Embedded Controller Handbook and MCS-96 data sheet for the actual device in use prior to the actual design.

For any user of an MCS-96 analog acquisition system (experienced or not), this document contains very useful information. It should be considered mandatory reading in addition to the latest Embedded Controller Handbook and MCS-96 data sheet for the actual device in use prior to the actual design.

MCS®-96 Analog Acquisition Primer

DAVID P. RYAN
INTEL CORPORATION

THE MCS-96 ANALOG ACQUISITION PRIMER

INTRODUCTION

As technology advances, embedded control applications continue to reduce chip-count and demand micro-controllers with increased features to assist system-cost reduction. Since every embedded control application interfaces with the physical world, and the physical world is an analog process, it was inevitable that microcontrollers would include integrated analog acquisition capabilities.

December 1987

It opened the door to cost reduction of high volume applications that required analog inputs. The device fit well into applications that needed processing of analog data. But this chip, with its 8-bit CPU, could not perform in high-end applications requiring analog inputs, or in applications that had computationally demanding analog tasks.

With the introduction of the MCS-96 family of 16-bit microcontrollers in 1983, the combined CPU and A/D performance became available to greatly reduce the system cost of mid- and high-performance embedded control applications. These are applications which were customarily implemented with 16-bit microcontrollers and were teamed with analog acquisition capabilities. There are less than a dozen 16-bit microcontrollers with an integrated analog acquisition system. For example, closed-loop servo control had been implemented almost exclusively by using analog methods. When an MCS-96 device is designed into such an application, it is not only replacing a microcontroller or microprocessor, but it also replaces closed-loop analog circuitry which never before came in contact with the digital system.

To take full advantage of this new level of integration, digital designers must become familiar with analog acquisition, and analog designers must become familiar

THE MCS®-96 ANALOG ACQUISITION PRIMER

INTRODUCTION

As technology advances, embedded control applications continue to reduce chip-count and demand microcontrollers with increased features to assist system-cost reduction. Since every embedded control application interfaces with the physical world, and the physical world is an analog process, it was inevitable that microcontrollers would include integrated analog acquisition capabilities.

The first such integration of standard microcontroller and A/D converter occurred on Intel's 8022 in 1978. This opened the door to cost reduction of high volume applications that required analog inputs. The device fit well into applications that needed processing of analog data. But this chip, with its 8-bit CPU, could not perform in high-end applications requiring analog inputs, or in applications that had computationally demanding analog tasks.

With the introduction of the MCS®-96 family of 16-bit microcontrollers in 1982, the combined CPU and A/D performance became available to greatly reduce the system cost of mid- and high-performance embedded control applications. These are applications which were customarily implemented with 16-bit microprocessor chip-sets teamed with analog acquisition chip sets.

There are less obvious avenues for system cost reduction when a 16-bit CPU is teamed with an on-chip analog acquisition system. For example, closed-loop servo control had been implemented almost exclusively by using analog methods. When an MCS-96 device is designed into such an application, it is not only replacing a microcontroller or microprocessor, but it also replaces closed-loop analog circuitry which never before came in contact with the digital system.

To take full advantage of this new level of integration, digital designers must become familiar with analog acquisition, and analog designers must become familiar

with digital methods of processing analog signals. This Application Note assists with the first task—understanding of an analog acquisition system.

Designers experienced with analog design, or analog acquisition systems, may find no revelations herein. To those unfamiliar with analog acquisition systems, this Ap Note provides a tutorial on the subject and will serve as a handy reference.

Answering the limitless number of analog circuit design questions is beyond the scope of this Ap Note. Suffice it to say that the effort placed on the design of analog circuits should increase with a decreasing error budget.

At a minimum, the applications literature of op-amp manufacturers and analog design manuals are a good place to start. Furthermore, the applications literature of monolithic analog acquisition system manufacturers should be consulted since the suggestions presented therein are largely transportable to any A/D system.

This Ap Note is organized in the following sections. The components of an analog acquisition system and the errors associated with each is first explained. Then, interfacing suggestions and ideas for getting more resolution are presented. Finally, a set of appendices provides back-up information, a bibliography, actual converter data and some program listings.

The definitions of terms used, and the examples presented, are drawn from the body of applications literature publicly available on the components of an analog acquisition system. There is usually no single meaning for a particular term or specification used to describe analog acquisition. However, there is, in most cases, a generally accepted definition which is most often used. To the extent possible, we have adopted the most used definition. To avoid any ambiguity, Appendix A lists the dictionary of terms as used to refer to the analog acquisition systems of MCS-96 devices.

For any users of an MCS-96 analog acquisition system (experienced or not), this document contains very useful information. It should be considered mandatory reading in addition to the latest Embedded Controller Handbook and MCS-96 data sheet for the actual device in use prior to the actual design.

WHAT IS AN ANALOG ACQUISITION SYSTEM?

An analog acquisition system is a collection of individual units which, when logically configured, form a system capable of converting an analog input to a digital value.

The typical components of an Analog Acquisition Unit (Figure 1) include an Analog-to-Digital Converter (A/D), a Sample-and-Hold (S/H) and an Analog Multiplexer (MUX). The A/D converts the infinitely varying analog voltage present on the S/H into a digital representation for use by the digital system. The S/H is required so a "snapshot" of a changing analog input can be stored for conversion by the A/D. The MUX is used to leverage the investment in the A/D by allowing a large number of isolated analog input channels to use the same converter.

The conversion result of an MCS-96 device is a 10-bit ratiometric representation of the input voltage. This produces a stair-stepped transfer function when the output code is plotted versus input voltage. See Figure 2.

The resulting digital codes can be taken as simple ratio-metric information, or they can be used to provide information about absolute voltages or relative voltage changes on the inputs. The more demanding the application is on the A/D converter, the more important it is to fully understand the converter's operation. For simple applications, knowing the absolute error of the converter is sufficient. However, controlling a closed loop with analog inputs necessitates a detailed understanding of an A/D converter's operation and errors.

The errors inherent in an analog-to-digital conversion process are many: quantizing error; zero offset; full-

scale error; differential non-linearity; and non-linearity. These are "transfer function" errors related to the A/D converter. In addition, the S/H and MUX may induce channel dissimilarities and sampling error (described later).

Fortunately, one "Absolute Error" specification is available which describes the sum total of all deviations between the actual conversion process and an ideal converter. The various sub-components of error are, however, important in many applications. These error components are described in Appendix A and in the text below where ideal and actual converters are compared.

A/D Converter

There are at least three well-recognized methods for converting an analog voltage to a digital value—flash, dual slope and successive approximation.

Flash A/Ds are the fastest, and most expensive converters for a given accuracy. Flash converters typically resolve bits of the result in parallel to achieve fast conversions. Flash converter speeds are measured in tens-of-nanoseconds.

Dual slope converters are the slowest, but most accurate. Dual slope conversion is rather insensitive to noise on the input, but conversion times are measured in milliseconds.

Successive approximation converters provide a balanced tradeoff between speed and accuracy. Successive approximation conversion times are measured in tens-of-microseconds, and converter implementations are very economical for a given accuracy.

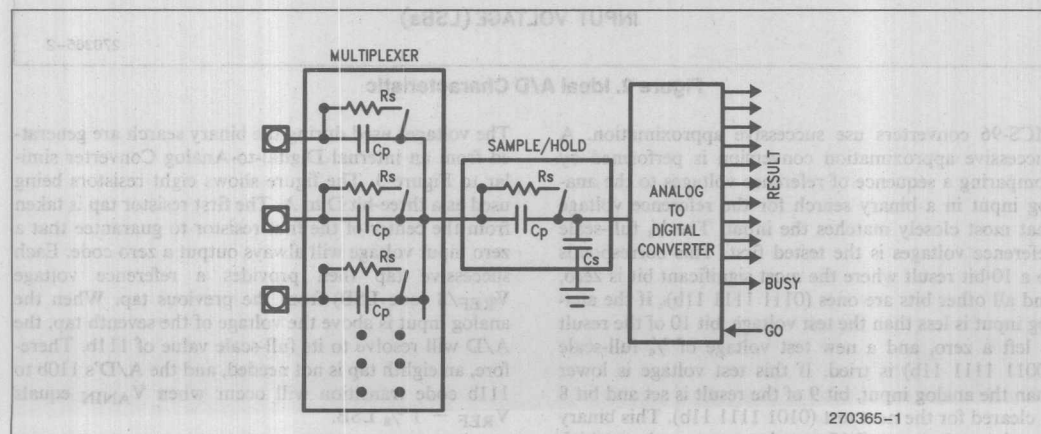


Figure 1. An Analog Acquisition System

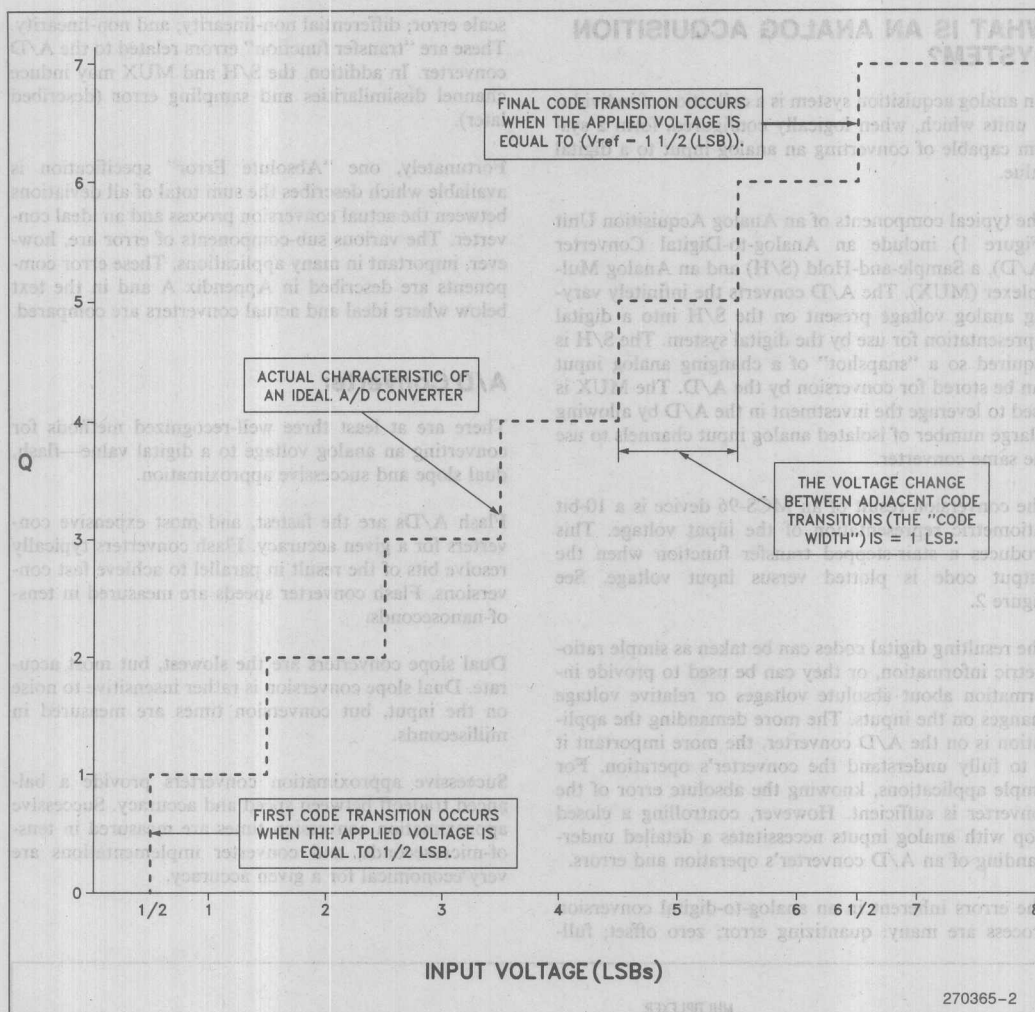


Figure 2. Ideal A/D Characteristic

MCS-96 converters use successive approximation. A successive approximation conversion is performed by comparing a sequence of reference voltages to the analog input in a binary search for the reference voltage that most closely matches the input. The $\frac{1}{2}$ full-scale reference voltages is the tested first. This corresponds to a 10-bit result where the most significant bit is zero, and all other bits are ones (0111 1111 11b). If the analog input is less than the test voltage, bit 10 of the result is left a zero, and a new test voltage of $\frac{1}{4}$ full-scale (0011 1111 11b) is tried. If this test voltage is lower than the analog input, bit 9 of the result is set and bit 8 is cleared for the next test (0101 1111 11b). This binary search continues until 10 tests have occurred, at which time the valid 10-bit conversion result resides in a register where it can be read by software.

The voltages used during the binary search are generated from an internal Digital-to-Analog Converter similar to Figure 3. The figure shows eight resistors being used as a three-bit D to A. The first resistor tap is taken from the center of the first resistor to guarantee that a zero input voltage will always output a zero code. Each successive tap then provides a reference voltage $V_{REF}/8$ (one LSB) from the previous tap. When the analog input is above the voltage of the seventh tap, the A/D will resolve to its full-scale value of 111b. Therefore, an eighth tap is not needed, and the A/D's 110b to 111b code transition will occur when V_{ANIN} equals $V_{REF} - 1 \frac{1}{2}$ LSB.

The first error seen in this process is unavoidable, and results from the conversion of a continuous voltage to

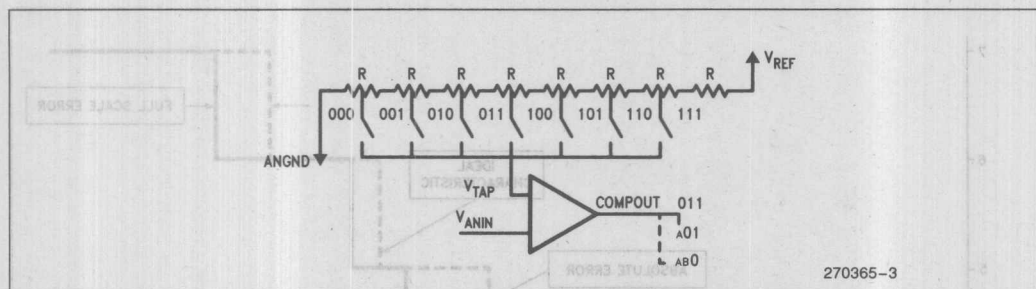


Figure 3. A Three-Bit D-to-A

an integer digital representation. This error is called quantizing error, and is always ± 0.5 LSB. Quantizing error is the only error seen in a perfect A/D converter, and is obviously present in actual converters. Figure 2 shows the transfer function for an ideal 3-bit A/D converter (i.e. the Ideal Characteristic).

Note that in Figure 2 the Ideal Characteristic possesses unique qualities: its first code transition occurs when the input voltage is 0.5 LSB; its full-scale code transition occurs when the input voltage equals the full-scale reference minus 1.5 LSB; and its code widths are all exactly one LSB. These qualities result in a digitization without offset, full-scale or linearity errors. In other words, a perfect conversion.

Figure 4 shows an Actual Characteristic of a hypothetical 3-bit converter which is not perfect. When the Ideal Characteristic is overlaid with the imperfect characteristic, the actual converter is seen to exhibit errors in the location of the first and final code transitions and code widths. The deviation of the first code transition from ideal is called "zero offset". The deviation of the final code transition from ideal is "full-scale error".

The deviation of the code widths from ideal causes two types of errors. Differential Non-Linearity and Non-Linearity. Differential Non-Linearity is a local linearity error measure, whereas Non-Linearity is an overall linearity error measure. For example, Figure 5a shows a transfer function with a large differential non-linearity and a little non-linearity. In contrast, Figure 5b shows a characteristic with small differential errors but a large overall linearity error.

Differential Non-Linearity is the degree to which actual code widths differ from the ideal width. Differential Non-Linearity gives the user a measure of how much the input voltage may have changed in order to produce a one count change in the conversion result.

If the absolute value of an input voltage is less important than the amount that the input changes, the differential non-linearity (DNL) specification of a converter is very important. For example, if the differential non-linearity of a converter is less than ± 0.5 LSB, a one count change in the digital result means that the input voltage changed at most 1.5 LSB (1 LSB ideal ± 0.5 LSB DNL). This is a much more accurate description of the input voltage change than would be available if the differential non-linearity of the converter was not known.

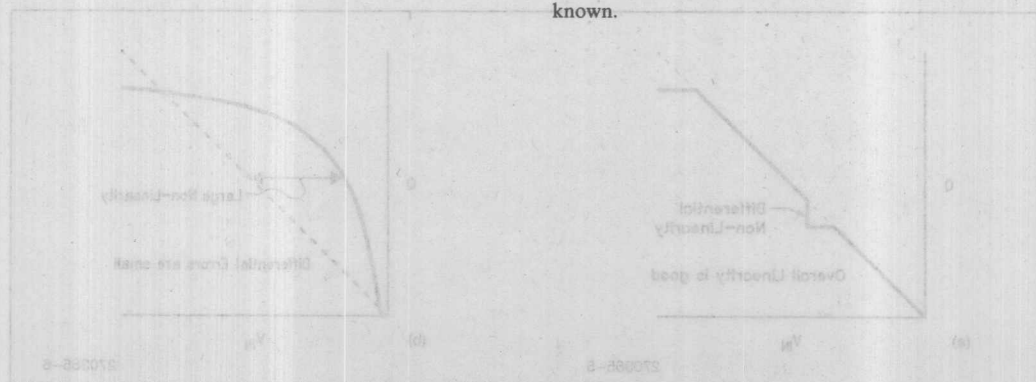


Figure 5. Types of Linearity Errors

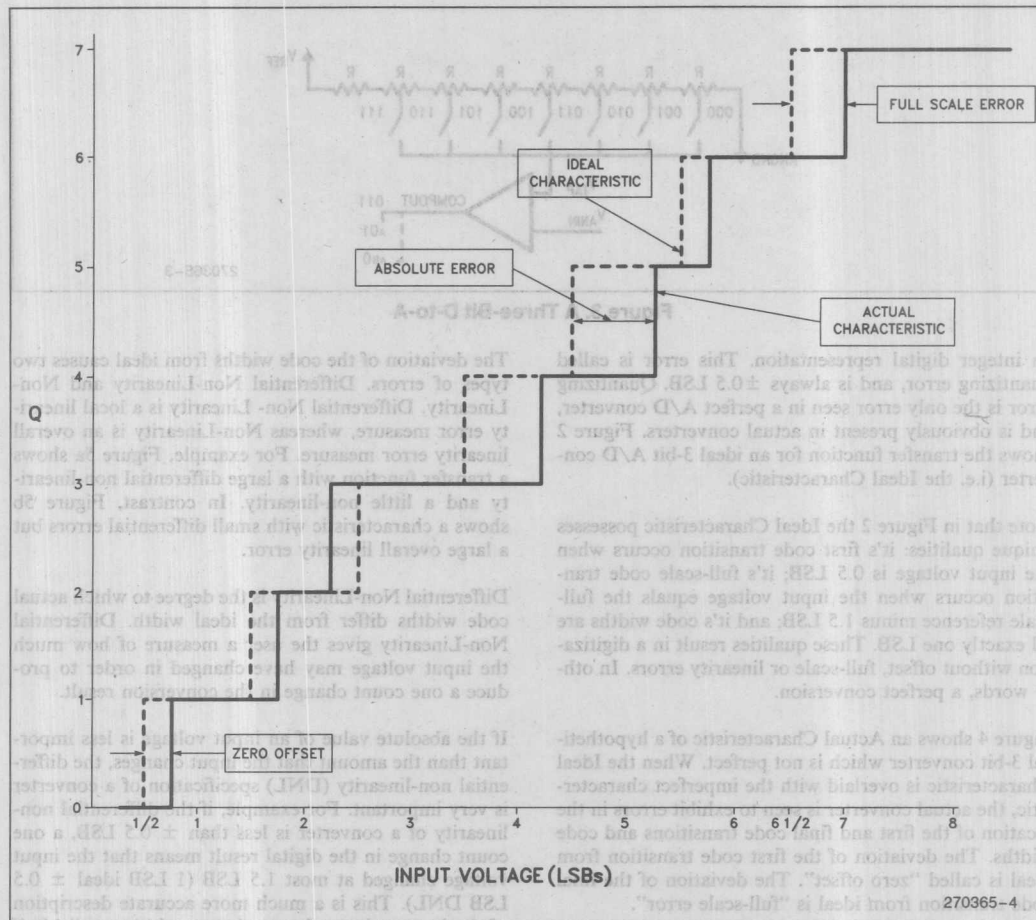


Figure 4. Actual and Ideal Characteristics

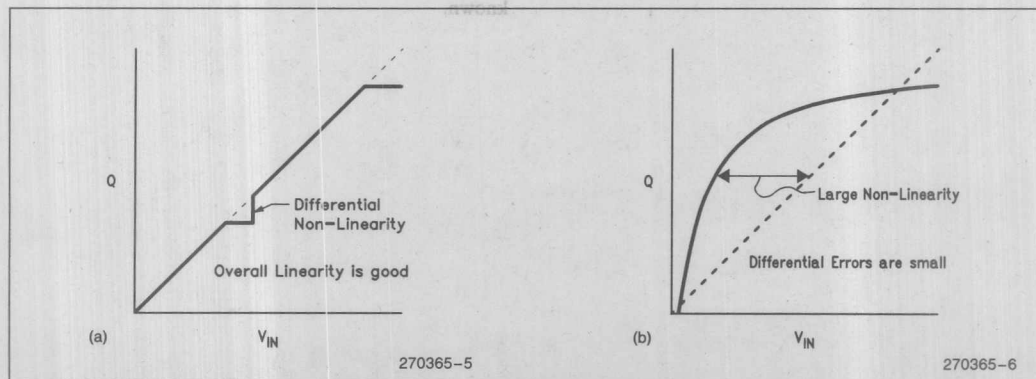


Figure 5. Types of Linearity Errors

Non-Linearity is the worst case deviation of code transitions from the corresponding code transitions of the Ideal Characteristic. Non-Linearity describes how much Differential Non-Linearities could add to produce an overall maximum departure from a linear characteristic.

If the Differential Non-Linearity errors are large enough, it is possible for an A/D converter to miss codes or exhibit non-monotonicity. Neither behavior is desirable in a closed-loop system. A converter has no missed codes if there exists for each output code a unique input voltage range that produces that code only. A converter is monotonic if every subsequent code change represents an input voltage change in the same direction. Figure 6a shows a converter with missed codes. Figure 6b shows a non-monotonic converter.

Differential Non-Linearity and Non-Linearity are quantified by measuring the Terminal Based Linearity Errors. A Terminal Based Characteristic results when an Actual Characteristic is shifted and scaled to eliminate zero offset and full-scale error (see Figure 7). The Terminal Based Characteristic is similar to the Actual Characteristic that would be seen if zero offset and full-scale error were externally trimmed away. In practice, this is done by using input circuits which include gain and offset trimming. (See the Application Hints section for more details.)

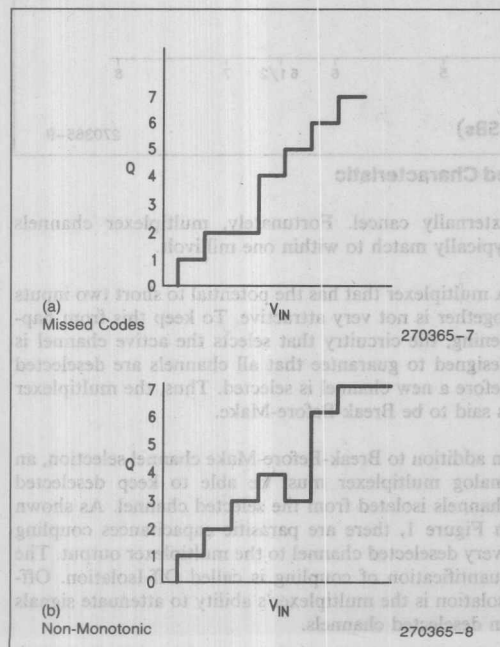


Figure 6. Undesirable Converter Operation

An often overlooked characteristic of A/D converters is that code transitions do not really occur instantaneously at some finite set of input voltages. Specific code transitions can be analyzed by doing repeated conversions around the transition point using a high accuracy input voltage. When this is done, we find that there is actually a range of voltages around code transitions where both the lower and upper codes occur for repeated conversions on the same input voltage.

Figure 8 shows this "repeatability" error. At the lower end of the region of repeatability error the lower code is most prevalent, but the upper code will occur in a small percentage of the conversion attempts. As the input voltage increases slightly, a point is reached where both lower and upper codes occur with 50 percent probability. As the input voltage moves slightly higher, the upper code occurs most often with the lower code showing up in a small percentage of conversions.

The repeatability error is due to the fundamental ability of the comparator in the A/D to resolve very similar voltages. Random noise also contributes to repeatability errors. On MCS-96 devices, the width of the region of repeatability error has been found to be typically 1 mV to 1.25 mV. Since this error is specified, all other errors are specified assuming the code transitions occur at the voltage where adjacent codes are equally likely.

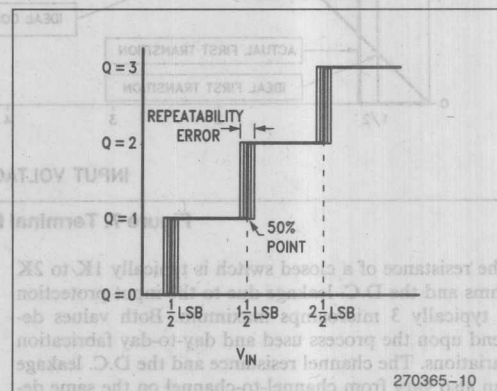


Figure 8. Repeatability Error

The Multiplexer

The eight channel multiplexer is implemented as a collection of eight MOS switches. Only one of eight can be closed at any instant in time. Figure 1 shows the multiplexer with the switches acting as resistors when closed and as small parasitic capacitors when open. The input protection devices on the analog input pins are also considered a part of the multiplexer.

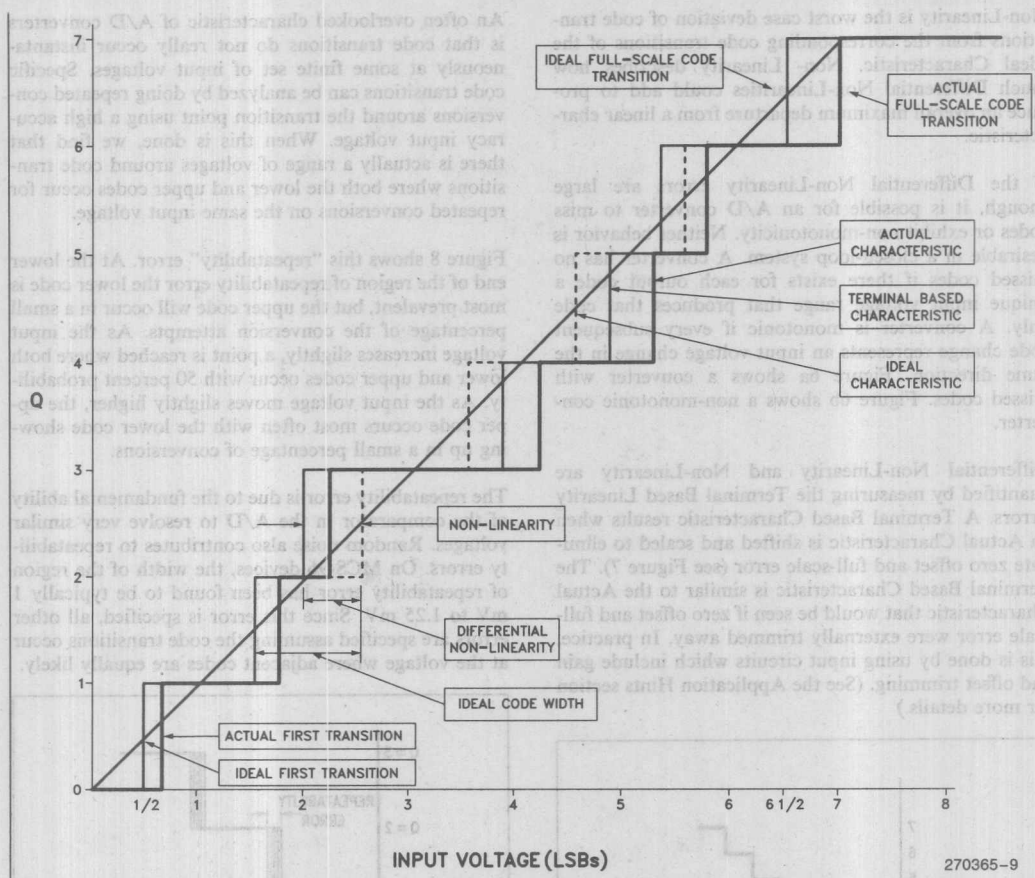


Figure 7. Terminal Based Characteristic

The resistance of a closed switch is typically 1K to 2K ohms and the D.C. leakage due to the input protection is typically 3 microamps maximum. Both values depend upon the process used and day-to-day fabrication variations. The channel resistance and the D.C. leakage can also vary from channel-to-channel on the same device. These variations can be seen in the conversion process and are described by the channel-to-channel matching specification.

Channel-to-channel matching specifies the input voltage differences induced by mismatched elements of the multiplexer. This error is quantified by measuring the difference between the input voltages necessary to cause the same code transition to occur through different multiplexer channels under identical test conditions.

Matching errors are more complex than a simple voltage offset between channels, and thus are difficult to

externally cancel. Fortunately, multiplexer channels typically match to within one millivolt.

A multiplexer that has the potential to short two inputs together is not very attractive. To keep this from happening, the circuitry that selects the active channel is designed to guarantee that all channels are deselected before a new channel is selected. Thus, the multiplexer is said to be Break-Before-Make.

In addition to Break-Before-Make channel selection, an analog multiplexer must be able to keep deselected channels isolated from the selected channel. As shown in Figure 1, there are parasitic capacitances coupling every deselected channel to the multiplexer output. The quantification of coupling is called Off-Isolation. Off-isolation is the multiplexer's ability to attenuate signals on deselected channels.

Sample-and-Hold

The sample-and-hold of an analog acquisition system can be built using an analog switch and a sample capacitor. As with the multiplexer, there is also a parasitic capacitance coupling the switch input to the sample capacitor when the switch is open (Figure 1).

The resistance of the sample-and-hold switch combines with the series resistance of the multiplexer to impede the current necessary to charge the sample capacitor. For example, with a 5K ohm total input resistance from the pin to the 2 pf sample capacitor, the RC time constant is 10 nS ($2 \text{ pf} \times 5\text{K ohms}$).

During the one microsecond that the sample capacitor is connected to the input, 100 time constants elapse (1 microsecond/10 nS). This means that the sample capacitor is 100 percent of the voltage on the input pin ($1 - e^{-100}$), assuming a zero source impedance.

If a source impedance of 2K ohms is assumed, the RC time constant of the sampling process would be 14nS ($7\text{K ohms} \times 2 \text{ pf}$). Thus, 71.4 time constants would pass in one microsecond resulting in the sample capacitor being charged to within 99.9 percent of its final value. Source impedances above 2K ohms would begin to degrade the conversion accuracy due to D.C. leakage (described later).

Figure 9 shows the actual input voltage and the sampled voltage approaching the input voltage. Once the sample-and-hold switch closes, the sample window begins. The sample window extends for four state times and ends with the sample-and-hold switch opening on MCS-96 devices (except 8X9X-90, which is 8 state times and has no sample-hold). Figure 9 also shows the sample delay, which is the delay from the time a start conversion signal is generated to the time a conversion process begins.

It is important to understand the uncertainties associated with the timing of the sample-and-hold. Digital signal processing algorithms rely upon the "spectral purity" of the sampling process. If the sample window jumps around with respect to the start conversion signal, or if the start conversion signal cannot be generated at precise times, consecutive samples of input data will not be equally spaced in time (i.e. sampling will be spectrally impure).

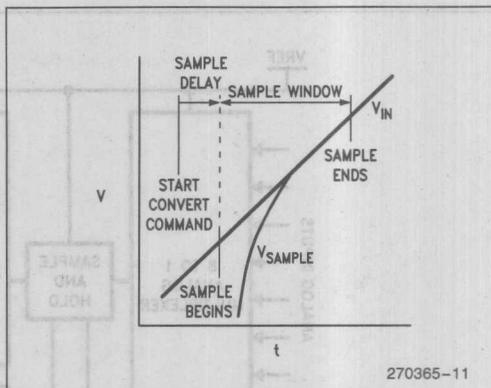


Figure 9. Sample-and-Hold Voltage

To improve the spectral purity of the sampling in digital signal processing applications, sequential MCS-96 start conversion signals can be generated with less than 50 nanoseconds of jitter using the HSO unit. The sample delay and sample time are also a constant number of state times to within 50 nanoseconds each.

Once the sample window closes, it is desired that all further changes on any input channel be isolated from the sample capacitor. The multiplexer's off-isolation is responsible for isolating deselected channels, while the sample-and-hold switch must attenuate changes on the selected channel. This source of error is described as Feedthrough. Feedthrough is quantified as the ability of the sample-and-hold to reject unwanted signals on its input.

Other factors that affect a real A/D Converter system include sensitivity to temperature. Temperature sensitivities are described by the change in typical specifications with a change in temperature.

The MCS®-96 Conversion Sequence

The MCS-96 Analog Acquisition System includes an eight channel analog multiplexer, sample-and-hold circuit and 10-bit analog to digital converter (Figure 10). An MCS-96 device can therefore select one of eight analog inputs, sample-and-hold the input voltage and convert the voltage into a digital value. Each conversion takes 22 microseconds (8097BH), including the time required for the sample-hold (with XTAL1 = 12 MHz). The method of conversion is successive approximation.

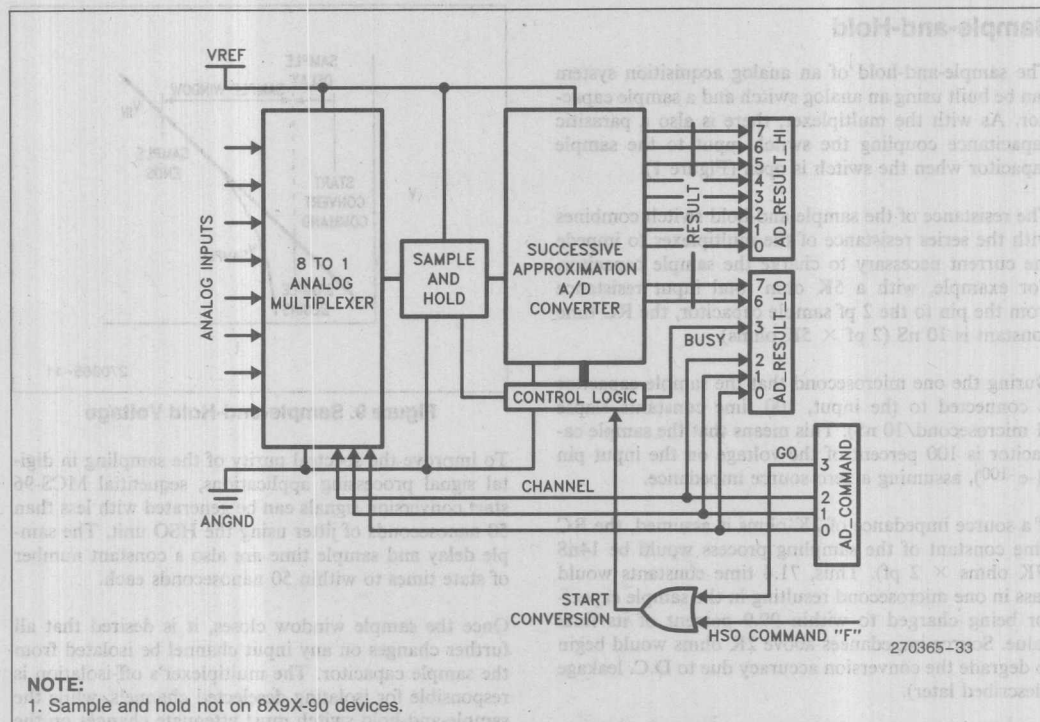


Figure 10. A/D Converter Block Diagram

The conversion process is initiated by the execution of HSO command OFH, or by writing a one to the GO Bit in the A/D Control Register. Either activity causes a start conversion signal to be sent to A/D control logic. If an HSO command was used, the conversion process will begin when Timer 1 increments. This aids applications attempting to approach spectrally pure sampling, since successive samples spaced by equal Timer 1 delays will occur with a variance of about ± 50 ns (assuming a stable clock on XTAL1). However, conversion initiated by writing a one to the ADCON register GO Bit will start within three state times after the instruction has completed execution, resulting in a variance of about $0.75 \mu\text{s}$ ($\text{XTAL1} = 12 \text{ MHz}$).

Once the A/D unit receives a start conversion signal, there is a one state time delay before sampling (sample delay) while the successive approximation register is reset and the proper multiplexer channel is selected. After the sample delay, the multiplexer output is connected to the sample capacitor and remains connected for four state times (sample time). After this four state time "sample window" closes, the input to the sample capacitor is disconnected from the multiplexer so that changes on the input pin will not alter the stored charge while

the conversion is in progress. The sample delay and sample time uncertainties are each approximately ± 50 ns, independent of clock speed.

To perform the actual analog-to-digital conversion the MCS-96 implements a successive approximation algorithm. The converter hardware consists of a 256-resistor ladder, a comparator, coupling capacitors and a 10-bit successive approximation register (SAR) with logic that guides the process. The resistor ladder provides 20 mV steps ($V_{\text{REF}} = 5.12\text{V}$), while capacitive coupling is used to create 5 mV steps within the 20 mV ladder voltages. Therefore, 1024 internal reference voltages are available for comparison against the analog input to generate a 10-bit conversion result. Appendix B contains a detailed description of the method used to generate 1024 voltages from a 256-resistor chain.

The total number of state times required for a 10-bit conversion varies from one MCS-96 version to the next. Attempting to short-cycle the 10-bit conversion process by reading A/D results before the done bit is set may work on some versions of MCS-96 devices, however it is not recommended. Short-cycling is not tested, nor is it guaranteed. Furthermore, it may not work on future MCS-96 devices.

APPLICATION HINTS

The analog signals that must be converted by an analog acquisition system vary widely. The analog input may arrive at the controller as a voltage or current. The range may be 0 to 1 volt or ± 30 volts, or some other arbitrary range. The input may be linear, logarithmic, non-linear, or perturbed in some bizarre fashion. Although interfacing to such signals could be considered an art form, some simple suggestions are contained in this section.

Analog Inputs

The external interface circuitry to an analog input is highly dependent upon the application, and can impact converter characteristics. In the external circuit's design, important factors such as input pin leakage, sample capacitor size and multiplexer series resistance from the input pin to the sample capacitor must be considered.

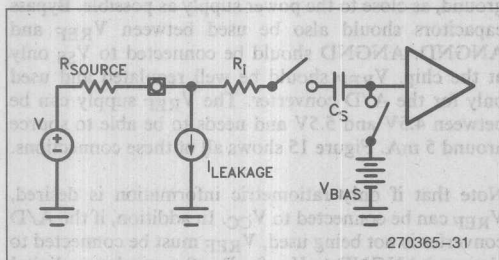


Figure 11. Idealized A/D Sampling Circuitry

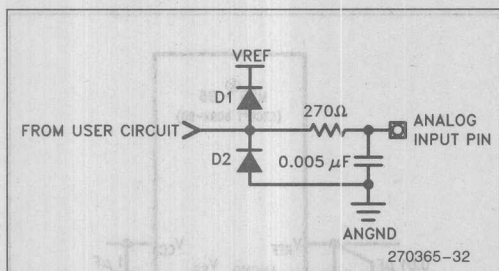


Figure 12. Suggested A/D Input Circuit

For the 8096BH, these factors are idealized in Figure 11. The external input circuit must be able to charge a sample capacitor (C_S) through a series of resistance (R_I) to an accurate voltage given a D.C. leakage (I_L). On the 8096BH, C_S is around 2 pF, R_I is around 5 K Ω and I_L is specified at 3 μ A maximum. In determining the source impedance R_S , V_{BIAS} is not important.

External circuits with source impedances of 1 K Ω or less will be able to maintain an input voltage within a

tolerance of about ± 0.61 LSB ($1.0 \text{ K}\Omega \times 3.0 \mu\text{A} = 3.0 \text{ mV}$) given the D.C. leakage. Source impedances above 2 K Ω can result in an external error of at least one LSB due to the voltage drop caused by the 3 μ A leakage. In addition, source impedances above 25 K Ω may degrade converter accuracy as a result of the internal sample capacitor not being fully charged during the 1 μ s (12 MHz clock) sample window.

Placing an external capacitor on each analog input will reduce the sensitivity to noise, as the capacitor combines with source resistance in the external circuit to form a low-pass filter. In practice, one should include a small series resistance prior to an external low leakage capacitor on the analog input pin and choose the largest capacitor value practical, given the frequency of the signal being converted. This provides a low-pass filter on the input, while the resistor will also limit input current during over-voltage conditions.

Figure 12 shows a simple analog interface circuit based upon the discussion above. The circuit in the figure also provides limited protection against over-voltage conditions on the analog input (limits to 2.6 mA with 270 Ω ($0.7/270$)). The circuit induces leakage from the diodes, which should be kept small.

The wide range of possible analog environments that must be interfaced to, or the existence of stringent accuracy requirements, makes the consideration of alternative input buffer configurations necessary. The most popular input buffer is a single op-amp in the non-inverting or inverting configurations of Figure 13.

In the non-inverting circuit of Figure 13 (a), the analog input is scaled by the buffer gain to output 5 volts when the input is at its maximum positive input. When the buffer input is 0 volts, the output will also be 0 volts.

In the inverting circuit of Figure 13 (b), a reference equal to the maximum possible input voltage is placed on the non-inverting input of the op-amp and the actual analog input is placed on the inverting input. The output voltage of the buffer is then proportional to the deviation of analog input from its maximum possible value. For example, when the analog input equals V_{MAX} , the buffer output will equal 0 zero volts. When the analog input equals its minimum value, the buffer output equals 5 volts. The digital result from the A/D converter might, of course, have to be complemented before being used.

The circuits of Figure 13 show only feedback resistors that set the gain of the buffer. In practice, it will often be necessary to include offset adjustments, gain trimming, temperature or frequency stability compensation, or components to build an active filter.

Figure 14 depicts a generalized non-inverting input buffer that offsets the analog input and scales the input

to a 5 volt range. The course offset is set by the ratio of R_{BIG1} and R_{BIG2} , while offset fine tuning is done by adjusting R_{TRIM} . The course gain is set by the ratio of R_{G1} and R_{G2} while gain trimming is done with R_{GTRIM} .

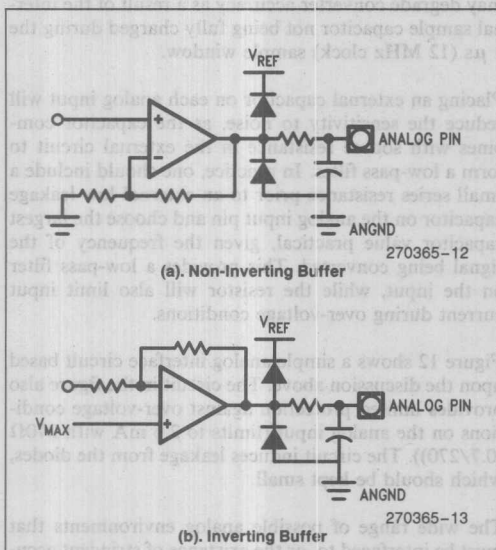


Figure 13

By trimming the offset and gain, not only can external component errors be zeroed out, but the offset and full scale error of the A/D converter can be nulled.

The procedure for nulling offset and gain is simple. First, a voltage is applied to V_{IN} which corresponds to the ideal first code transition of the A/D. R_{TRIM} is adjusted so that 50 percent of the conversion results are 0 while 50 percent are 1. Second, a voltage is applied to V_{IN} which corresponds to the ideal final code transition of the A/D converter. R_{GTRIM} is then adjusted until 50 percent of the conversion results are 3FEH and 50 percent are 3FFH. Once this adjustment is complete, the converter zero offset and full-scale errors are nulled, and could be ignored (except for temperature variation). This allows the system to rely upon the tighter, more descriptive converter specifications for Terminal Based Non-Linearity and Differential Non-Linearity.

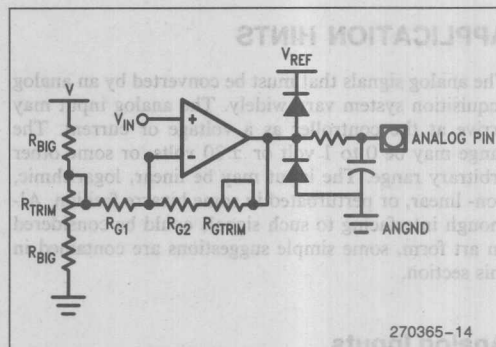


Figure 14. Trimming Offset and Gain

Analog References

Reference supply levels strongly influence the absolute accuracy of the conversion. For this reason, it is recommended that the ANGND pin be tied to a clean ground, as close to the power supply as possible. Bypass capacitors should also be used between V_{REF} and ANGND. ANGND should be connected to V_{SS} only at the chip. V_{REF} should be well regulated and used only for the A/D converter. The V_{REF} supply can be between 4.5V and 5.5V and needs to be able to source around 5 mA. Figure 15 shows all of these connections.

Note that if only ratiometric information is desired, V_{REF} can be connected to V_{CC} . In addition, if the A/D converter is not being used, V_{REF} must be connected to V_{CC} and ANGND to V_{SS} for Port0 to work as a digital port.

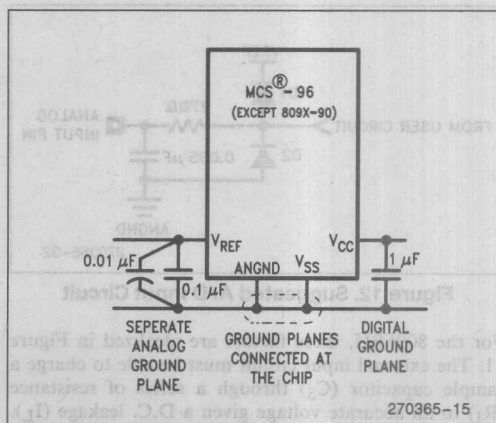


Figure 15. Supply Decoupling

Getting More Resolution

Given that the A/D converter can convert an analog input ranging from 0 volts to 5 volts into 1024 steps of 5 millivolts each, the desire for more resolution can come from three basic needs – need extra LSB, need extra MSB, need BOTH.

The configuration shown in Figure 16 can be used to solve each of the “more resolution” problems. This setup requires the use of two input channels with different offsets and gains.

When the 5 millivolt step size of the A/D is too large for the application requirements, but the 5 volt range is sufficient, the system needs an “extra LSB”. For example, an application requiring 2.5 millivolt steps over a 5 volt range needs an 11-bit conversion result. The 11th bit needs to be added to the least significant side of the 10-bit result (the “right”). This can be achieved using the circuit of Figure 16.

If both channels are set for a gain of 2, with channel 1 offset to 2.5 volts, the 5 volt input range is split into 2.5 volt ranges that are amplified by two before being input to the A/D. While V_{IN} is between 0 and 2.5 volts, channel 0 will be providing a proportional voltage between 0 volts and 5 volts to the A/D converter. Channel 1 will be clamped to 5 volts. When V_{IN} rises above 2.5 volts, channel 1 will begin to output a proportional voltage between 0 volts and 5 volts to the A/D converter and channel 0 will be clamped at 5 volts. Using this method, an 11-bit (2048 step) result is created with 2.5 millivolt steps (i.e. an extra LSB).

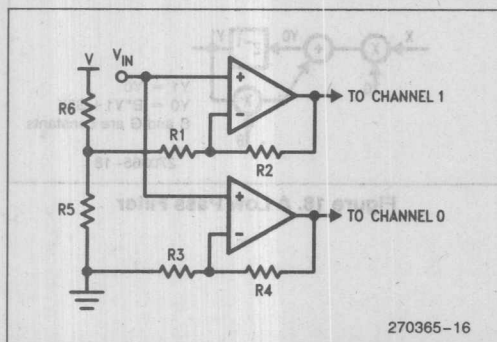


Figure 16. A Flexible Input Circuit

It is useful to note that only one conversion per sample will be required if the software keeps track of which channel is active. The only time that two conversions will be required for one sample is when the voltage crosses the midpoint.

The second reason that “more resolution” is requested is the need for an “extra MSB”. When the converter’s input voltage range is too small (5 volts when 10 volts is needed), but 5 millivolt steps over the actual input voltage range is sufficient, an extra bit is needed on the most significant (“left”) side of the 10-bit result. The circuit of Figure 16 can also be used, with different gains and offsets, to satisfy this extra MSB need by splitting the 10 volt range into 5 volt ranges.

If both channels of Figure 16 are set for unity gain, and channel 1 is offset to 5 volts, an 11-bit conversion result with 5 millivolt steps is available. While V_{IN} is in the lower half of its range (0 volts to 5 volts), channel 0 will be active. While V_{IN} is in the upper half of its range (5 volts to 10 volts), channel 1 will be active. Thus, an extra MSB is created.

For applications requiring multiple extra bits of result, the solutions can become more “elegant” (i.e. elaborate). However, it is profitable to first squeeze the most out of the now familiar circuit in Figure 16.

Assume that the analog input, V_{IN} , ranges from 0 volts to 10 volts, and it is desired to measure this range in 2.5 millivolt steps. This requires two extra bits of result – one extra MSB and one extra LSB. A simple extrapolation of the preceding discussion of creating extra bits might have the designer planning to tie up four channels of the multiplexer needlessly. Needless, that is, if the application is a typical control application where the high accuracy requirements are only important in the “normal” operating range of the process. Outside of the normal operating range is the “possible” operating range which must be measured, but with less stringent requirements.

Since the requirements of the normal range set the necessary LSB weight, and the extent of the possible range sets the maximum voltage span, it follows that only two channels need to be used (Figure 16). Channel 0 would be set with a gain that compressed the possible V_{IN} range to 5 volts, while channel 1 would be offset to the normal operating range and would have a gain of two to expand this region of critical interest. With this ap-

proach, 100 percent of the normal operating range is digitized in 2.5 millivolt steps, while 100 percent of the possible range is digitized in 10 millivolt steps.

Unfortunately, not all high resolution applications can be described as a process with a small region of in-control operation, where the process is out-of-control outside of that small region. For example, it is necessary to measure airflow in an engine controlling carburetion. The air flow at idle is likely to be several orders-of-magnitude lower than the airflow at full RPM. The process needs to be in tight control over the entire range, not only when the engine is at half-speed.

When it is desired to measure a process with a fixed percent of error throughout a range spanning several orders-of-magnitude, a non-linear input buffer becomes attractive. For example, assume that the analog signal that needs to be digitized can vary from 1 millivolt to 25 volts and describes a physical process that must be represented digitally with 1 percent error at any point in the possible input range. A linear solution to this application would require a converter with a 10 microvolt LSB ($1\% \times 1 \text{ mV}$), and a resolution of 22 bits ($25 \text{ V}/10 \text{ microvolts}$). This is clearly undesirable.

The use of a log input buffer to compress the 25 volt range logarithmically to 5 volts would satisfy the application requirements. The input would range from 1 millivolt to 25 volts with the output ranging from 0 volts to 5 volts proportionally to the log of $V_{IN}/1\text{mV}$. Each one-percent change in the input voltage would change the output voltage by 5 millivolts (one count). The antilog could be taken in software using a lookup table, or the control calculations could be performed in a log base.

Simple inexpensive log-amps can be built as in Figure 17, or high-accuracy, self-contained log-amps can be purchased. Which is chosen depends upon the application tradeoffs of price and performance.

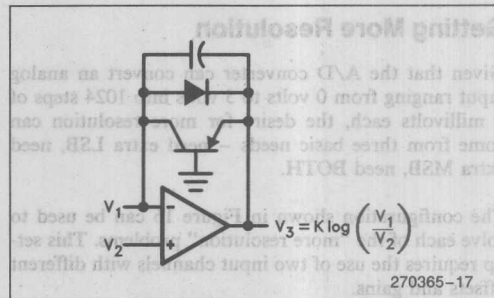


Figure 17. A Low-Cost Log Amplifier

Other techniques become available for consideration in systems that have slow sample rate requirements, but very high resolution requirements. In addition to the methods described above, which require external hardware, software filtering or other post-processing of the conversion results can be productive. Each method relies upon the ability to sample the analog input much faster than the system requires an analog input.

When resolution is limited by filterable noise, perhaps the most straightforward approach to post-processing is to oversample the input by a factor of N and digitally low-pass filter the data (i.e. weighted rolling average). A result would be reported to the rest of the system every N samples (Figure 18). A low-pass filter can increase the signal-to-noise ratio (SNR) by a factor of N (see bibliography). However, care must be taken to be certain that the input voltage varies slowly with respect to the sampling rate.

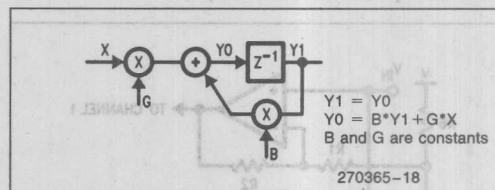


Figure 18. A Low Pass Filter

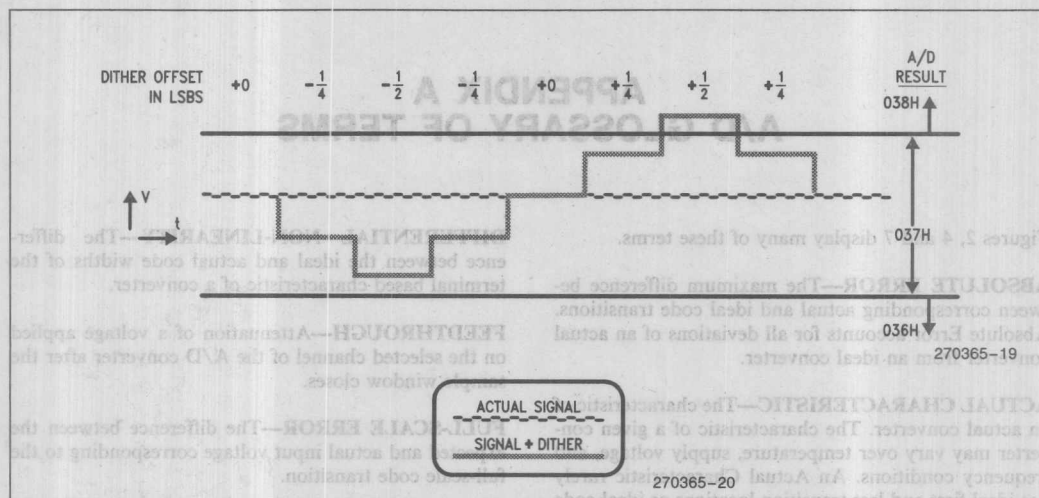


Figure 19. Dither

Another approach to creating more resolution is called "synchronized dither". Figure 19 shows an input voltage that is constant somewhere between two code transition points. This input is "dithered" by adding a small periodic waveform ($1/4$ LSB steps) to the input while performing an A/D conversion synchronized with each dither step. Every time the dither completes a full cycle, the eight conversion results are averaged to form one digitized value. Since the dither is periodic and symmetrical about 0 volts, its average impact on the input voltage is 0 volts.

The creation of extra resolution can be seen with the example shown in Figure 19. Without dither, the input voltage would always convert to 37H. With dither, one-eighth of the conversions would be 38H and $7/8$ of the conversions would be 37H. If every eight conversions were averaged, the result would be $37H + 1/8$ LSB. The possible results given a four level dither, where the input voltage was always within the 37H code width, would be

$$\begin{aligned} 36H + 5/8 \\ 36H + 7/8 \\ 37H + 0 \\ 37H + 1/8 \\ 37H + 3/8 \end{aligned}$$

Hence, four new levels exist (two bits).

Dither will only create more resolution up to the limit of the A/D converter comparator's ability to distinguish voltages. Since MCS-96 converter repeatability error is typically around 1 millivolt to 1.25 millivolts, $1/4$ LSB dither is the practical limit if no other processing is done. Figure 20 shows a simple method by which

the input voltage could be dithered under software control.

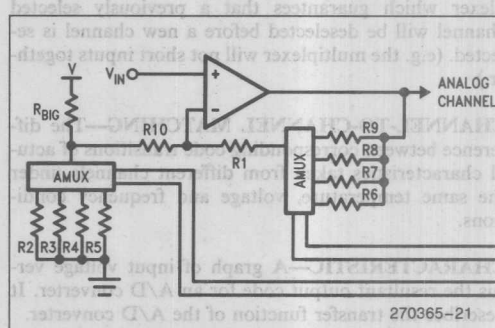


Figure 20. Software Controlled Offset and Gain

While only a few of the more obvious interfacing techniques were described here, there are as many innovative interfacing tricks as there are designers.

CONCLUSION

This application note provides a fundamental understanding of MCS-96 analog acquisition for the digital designer. Since answering the limitless number of analog circuit design questions is beyond the scope of this document, it is expected that analog design manuals and the large body of publicly available applications literature will be consulted for detailed design hints. Furthermore, the applications literature of monolithic analog acquisition system manufacturers should be consulted since the suggestions presented therein are largely transportable to any A/D system.

APPENDIX A A/D GLOSSARY OF TERMS

Figures 2, 4 and 7 display many of these terms.

ABSOLUTE ERROR—The maximum difference between corresponding actual and ideal code transitions. Absolute Error accounts for all deviations of an actual converter from an ideal converter.

ACTUAL CHARACTERISTIC—The characteristic of an actual converter. The characteristic of a given converter may vary over temperature, supply voltage, and frequency conditions. An Actual Characteristic rarely has ideal first and last transition locations or ideal code widths. It may even vary over multiple conversions under the same conditions.

BREAK-BEFORE-MAKE—The property of a multiplexer which guarantees that a previously selected channel will be deselected before a new channel is selected. (e.g. the multiplexer will not short inputs together.)

CHANNEL-TO-CHANNEL MATCHING—The difference between corresponding code transitions of actual characteristics taken from different channels under the same temperature, voltage and frequency conditions.

CHARACTERISTIC—A graph of input voltage versus the resultant output code for an A/D converter. It describes the transfer function of the A/D converter.

CODE—The digital value output by the converter.

CODE CENTER—The voltage corresponding to the midpoint between two adjacent code transitions.

CODE TRANSITION—The point at which the converter changes from an output code of Q , to a code of $Q + 1$. The input voltage corresponding to a code transition is defined to be that voltage which is equally likely to produce either of two adjacent codes.

CODE WIDTH—The voltage corresponding to the difference between two adjacent code transitions.

CROSSTALK—See "Off-Isolation".

D.C. INPUT LEAKAGE—D.C. Leakage current of an analog input pin.

DIFFERENTIAL NON-LINEARITY—The difference between the ideal and actual code widths of the terminal based characteristic of a converter.

FEEDTHROUGH—Attenuation of a voltage applied on the selected channel of the A/D converter after the sample window closes.

FULL-SCALE ERROR—The difference between the expected and actual input voltage corresponding to the full-scale code transition.

IDEAL CHARACTERISTIC—A characteristic with its first code transition at $V_{IN} = 0.5 \text{ LSB}$, its last code transition at $V_{IN} = (V_{REF} - 1.5 \text{ LSB})$ and all code widths equal to one LSB.

INPUT RESISTANCE—The effective series resistance from the analog input pin to the sample capacitor.

LSB - LEAST SIGNIFICANT BIT—The voltage value corresponding to the full-scale voltage divided by 2^n , where n is the number of bits of resolution of the converter. For a 10-bit converter with a reference voltage of 5.12 volts, one LSB is 5.0 mV. Note that this is different than digital LSBs, since an uncertainty of two LSBs, when referring to an A/D converter, equals 10 mV. (This has been confused with an uncertainty of two digital bits, which would mean four counts, or 20 mV.)

MONOTONIC—The property of successive approximation converters which guarantees that increasing input voltages produce adjacent codes of increasing value, and that decreasing input voltages produce adjacent codes of decreasing value.

NO MISSED CODES—For each and every output code, there exists a unique input voltage range which produces that code only.

NON-LINEARITY—The maximum deviation of code transitions of the terminal based characteristic from the corresponding code transitions of the actual characteristic of a converter.

OFF-ISOLATION—Attenuation of a voltage applied on a deselected channel of the A/D converter. (Also referred to as Crosstalk.)

REPEATABILITY—The difference between corresponding code transitions from different actual characteristics taken from the same converter on the same channel at the same temperature, voltage and frequency conditions.

RESOLUTION—The number of input voltage levels that the converter can unambiguously distinguish between. Also defines the number of useful bits of information which the converter can return.

SAMPLE DELAY—The delay from receiving the start conversion signal to when the sample window opens.

SAMPLE DELAY UNCERTAINTY—The variation in the Sample Delay.

SAMPLE TIME—The time that the sample window is open.

SAMPLE TIME UNCERTAINTY—The variation in the sample time.

During the sample window (Figure B1a), V_{ANIN} and V_{OP2} control the amount of charge stored in C_A and C_B (V_{OP2} controls the converter offset). Once the sample window closes (Figure B1b), voltages applied to V_{IN} and V_{IN2} will add or subtract charge proportional to $(V_{\text{ANIN}} - V_{\text{IN}})$ on C_A and $(V_{\text{OP2}} - V_{\text{IN2}})$ on C_B . Unless a voltage is applied to V_{IN} and V_{IN2} , the inverting comparator input of Figure B1b will remain at V_{BIAS} due to the charges on C_A and C_B . The non-inverting comparator input will always remain at V_{BIAS} and serves as a reference.

If a V_{IN} V_{IN2} combination is applied which causes the non-inverting input to drop below V_{BIAS} , the comparator will output a 1 to indicate that the applied voltage was lower than the original V_{ANIN} . To better understand how the circuit works, Figure B2 shows the superposition analysis used to form the equation for V_{OUT} . Given initial charges on C_A and C_B and new input voltages V_{IN} and V_{IN2} .

SAMPLE WINDOW—Begins when the sample capacitor is attached to a selected channel and ends when the sample capacitor is disconnected from the selected channel.

SUCCESSIVE APPROXIMATION—An A/D conversion method which uses a binary search to arrive at the best digital representation of an analog input.

TEMPERATURE COEFFICIENTS—Change in the stated variable per degree centigrade temperature change. Temperature coefficients are added to the typical values of a specification to see the effect of temperature drift.

TERMINAL BASED CHARACTERISTIC—An Actual Characteristic which has been rotated and translated to remove zero offset and full-scale error.

V_{CC} REJECTION—Attenuation of noise on the V_{CC} line to the A/D converter.

ZERO OFFSET—The difference between the expected and actual input voltage corresponding to the first code transition.

Before beginning a detailed description of the capacitive part of the conversion process, it is necessary to understand a few details about the resistor chain.

There are 320 resistors connected in series from the analog reference to analog ground. The actual value of the resistors only impacts the current through the resistor chain. If every resistor in the chain is the same value, the converter will function properly.

To reduce resistor-to-resistor variation, the chain is folded in half and then in an accordion fashion to provide a 16×16 block of resistors. This minimizes the sensitivity of the array to processing gradients, while also allowing the array to be addressed roughly similar to a 16×16 memory array.

APPENDIX B CAPACITIVE INTERPOLATION

A successive approximation A/D converter needs an internal D/A converter of the same resolution as the desired A/D result. A 10-bit D/A could have been made using a string of 1024 resistors connected from the analog reference at one end to ground at the other end. Although this would be technically ideal, such a circuit would be enormous. Therefore, a method was developed to generate the needed reference voltages using a small area of silicon so that an on-chip 10-bit A/D converter would be economical.

The method used relies upon a 256-resistor chain to generate reference voltages in 20mV (5.12V/256) steps while two ratioed capacitors are used to capacitively "interpolate" voltages in-between the resistor tap voltages. The area of the 256-resistor chain together with the capacitors is one-fourth the area of the would-be 1024 resistor chain.

Before beginning a detailed description of the capacitive part of the conversion process, it is necessary to understand a few details about the resistor chain.

There are 256 resistors connected in series from the analog reference to analog ground. The actual value of the resistors only impacts the current through the reference pin. If every resistor in the chain is the same value the converter will function properly.

To reduce resistor-to-resistor variation, the chain is folded in half, and then in an accordion fashion to produce a 16×16 block of resistors. This minimizes the sensitivity of the array to processing gradients, while also allowing the array to be addressed roughly similar to a 16×16 memory array.

As explained earlier, it is desired for the A/D converter to have its first code transition at $\frac{1}{2}$ LSB followed by subsequent code widths 1 LSB wide.

To accomplish this, each resistor is tapped in its center rather than between resistors. For example, the first resistor tap is half-way up the first resistor. This means that the zero resistor tap will output 10mV (20mV/2). When calculating the voltage on a certain resistor tap, you must add 10mV to the product of the tap number and 20mV.

The internal connections while an analog input is being sampled are shown in Figure B1a. Once sampling is complete, the analog input is disconnected and the comparator inputs are no longer clamped to V_{BIAS} (Figure B1b).

During the sample window (Figure B1a), V_{ANIN} and V_{OFS} control the amount of charge stored in C_A and C_B (V_{OFS} controls the converter offset). Once the sample window closes (Figure B1b), voltages applied to V_{IN} and V_{IN2} will add or subtract charge proportional to $(V_{ANIN} - V_{IN})$ on C_A and $(V_{OFS} - V_{IN2})$ on C_B . Unless a voltage is applied to V_{IN} and V_{IN2} . The inverting comparator input of Figure B1b will remain at V_{BIAS} due to the charges on C_A and C_B . The non-inverting comparator input will always remain at V_{BIAS} and serves as a reference.

If a V_{IN} , V_{IN2} combination is applied which causes the non-inverting input to drop below V_{BIAS} the comparator will output to a 1 to indicate that the applied voltage was lower than the original V_{ANIN} . To better understand how the circuit works, Figure B2 shows the superposition analysis used to form the equation for V_{OUT} , given initial charge on C_A and C_B and new input voltages V_{IN} and V_{IN2} .

Adding the independent effects shown in Figure B2 we have:

$$V_{OUT} = V_1 + V_2 + V_3 + V_4$$

$$V_{OUT} = V_{IN} \left(\frac{C_A}{C_A + C_B} \right) + V_{IN2} \left(\frac{C_B}{C_A + C_B} \right) + V_{AI} \left(\frac{C_A}{C_A + C_B} \right) + V_{BI} \left(\frac{C_B}{C_A + C_B} \right)$$

$$V_{OUT} = (V_{IN} + V_{AI}) \frac{C_A}{C_A + C_B} + (V_{IN2} + V_{BI}) \frac{C_B}{C_A + C_B}$$

The initial conditions on C_A and C_B are set-up as shown in Figure B3.

We can see that:

$$V_{AI} = V_{BIAS} - V_{ANIN} \quad (II)$$

$$V_{BI} = V_{BIAS} - V_{OFS} \quad (III)$$

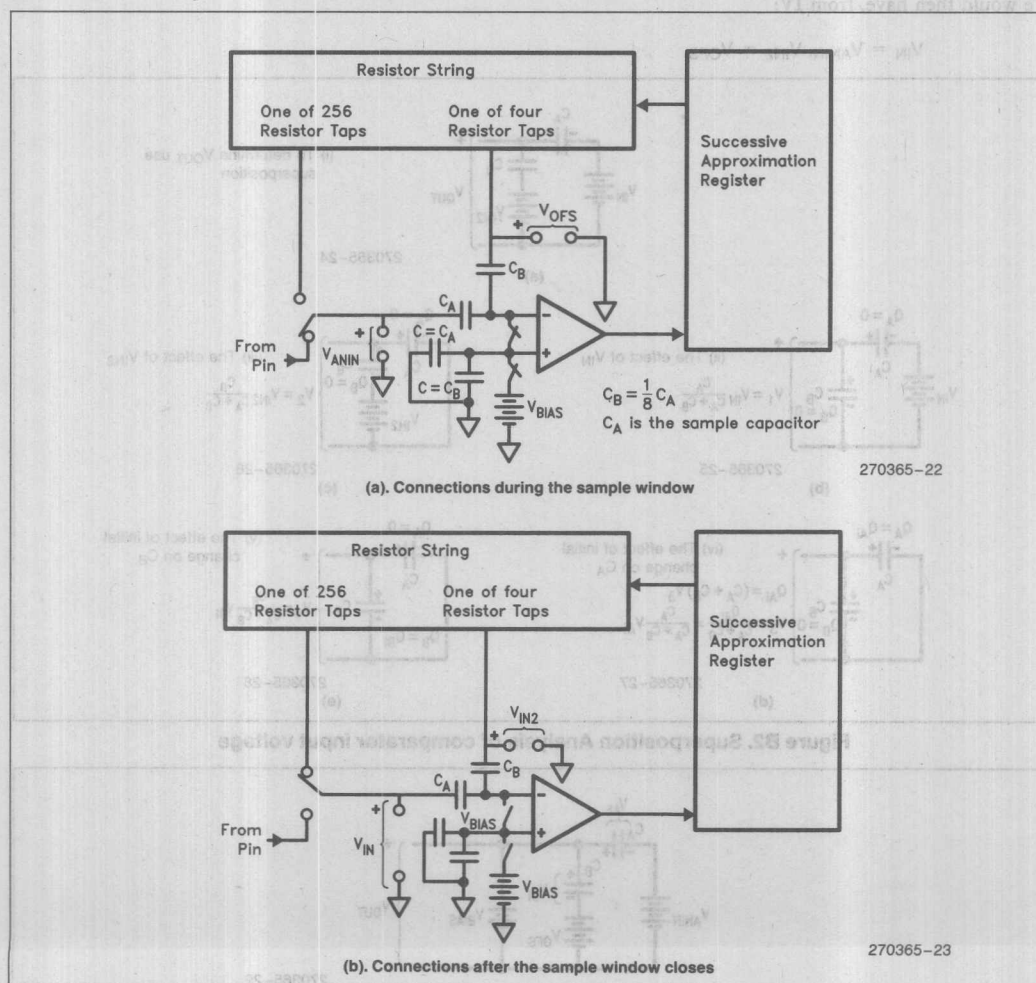


Figure B1

Substituting II and III into I we get:

$$V_{OUT} = (V_{IN} + V_{BIAS} - V_{ANIN}) \frac{C_A}{C_A + C_B} + (V_{IN2} + V_{BIAS} - V_{OFS}) \frac{C_B}{C_A + C_B} \quad (IV)$$

V_{OUT} becomes the input voltage to the comparator which ideally presents no load. The only way to make V_{OUT} approach the value of V_{BIAS} (after V_{BIAS} is removed) is to apply a voltage combination which makes equation IV evaluate to V_{BIAS} . If we had an infinitely variable internal voltage reference to use, we could just set the reference on V_{IN} to the value of V_{ANIN} and make $V_{IN2} = V_{OFS}$.

We would then have, from IV:

$$V_{IN} = V_{ANIN}, V_{IN2} = V_{OFS}$$

However, using a 256-resistor chain to provide references, we can find a V_{IN} , V_{IN2} combination which can bring V_{OUT} close to the value of V_{BIAS} . The 256-resistor chain provides a reference voltage in 20 mV steps. We can then take separate taps of the resistor chain and connect them to V_{IN} and V_{IN2} . The voltage attached to V_{IN} will couple to V_{OUT} by a factor of $C_A/(C_A + C_B) = 8/9$ from EQN IV. The voltage attached to V_{IN2} will couple to V_{OUT} by a factor of $C_B/(C_A + C_B)$. The ratio of the impacts on V_{OUT} of V_{IN} versus V_{IN2} is:

$$\left(\frac{\partial V_{OUT}}{\partial V_{IN}} \right) \div \left(\frac{\partial V_{OUT}}{\partial V_{IN2}} \right) = (8/9)/(1/9) = 8$$

Therefore, a voltage change on V_{IN} will affect the voltage seen at V_{OUT} eight times more than the same change placed on V_{IN2} .

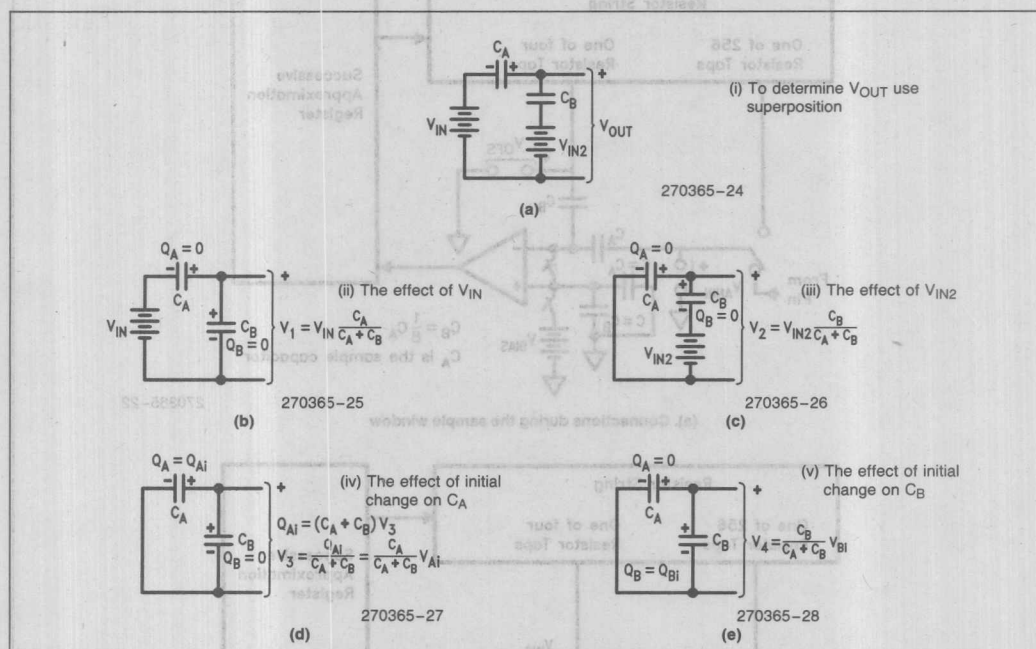


Figure B2. Superposition Analysis of comparator input voltage

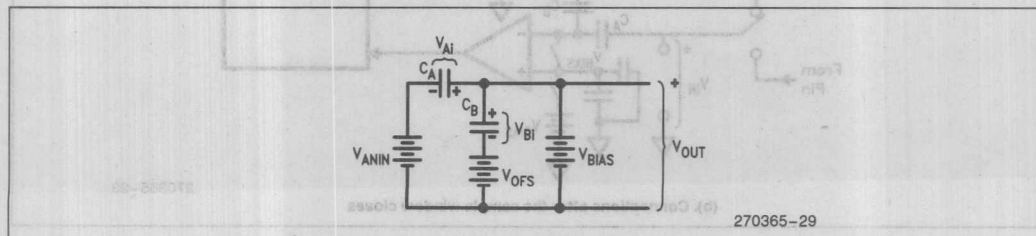


Figure B3. Initial Conditions

For example, assume the actual input voltage V_{ANIN} was 2.50mV during the sample window. Using EQN IV, and assuming $V_{BIAS} = 3V$ and $V_{OFS} = 70mV$, we substitute and find:

$$V_{OUT} = (V_{IN} + 2.9975) \times (8/9) + (V_{IN2} + 2.93) \times (1/9) \quad (V)$$

Using successive approximation, the first trial input voltage attempted corresponds to the digital code 0111 1111 11b ($127 \times 20mV + 10mV$). This means that the voltage applied to V_{IN} will be the 0111 1111b tap and the voltage applied to V_{IN2} will be the 0110b tap ($6 \times 20mV + 10mV = 3 \text{ LSB}$). Substituting these values into EQN V we have:

$$V_{OUT} = (2.550 + 2.9975) \times (8/9) + (0.130 + 2.93) \times (1/9) \quad (V)$$

$$V_{OUT} = 4.931 + 0.34 = 5.271 \quad (V)$$

Since the 3V reference is lower than V_{OUT} with these inputs, the comparator will output a 0 which is placed in the MSB of the successive approximation register. The next most significant bit of the SAR is then zero'd

and the new ladder tap applied to V_{IN} . The result of this second comparison, and the subsequent comparisons are shown in Table B1. The C program used to generate Table B1 is listed in Listing B1.

The value selected for V_{OFS} during the sample window may not be obvious. The purpose of V_{OFS} is to inject a constant offset in the sampling process so that the converter's first code transition will occur at 2.5mV.

Using EQN IV we can quickly see why V_{OFS} is chosen to be the fourth resistor tap ($4 \times 20mV + 10mV = 70mV$). For $V_{ANIN} = 2.5mV$, we want V_{OUT} to evaluate to V_{BIAS} when the SAR is OH.

$$V_{OUT} = \{(0.20 \text{ mV} + 10 \text{ mV}) + (V_{BIAS} - 2.5 \text{ mV})\} \times (8/9) + \{(0.20 \text{ mV} + 10 \text{ mV}) + (V_{BIAS} - 70 \text{ mV})\} \times (1/9)$$

$$V_{OUT} - V_{BIAS} = 7.5 \text{ mV} \times (8/9) - 60 \text{ mV} \times (1/9) = 0$$

Therefore, if $V_{OFS} = 70 \text{ mV}$, the converter's first code transition will be when $V_{ANIN} = 2.5 \text{ mV}$.

Table B1. Conversion Simulation

```
A to D simulator. (center taps) .. With
VIN = 0.002500
VCENT = 3.000000 VOFF = 0.070000
SAR = 1FFF ( 511) VOUT = 5.271111
SAR = FFE ( 255) VOUT = 4.133333
SAR = 7FE ( 127) VOUT = 3.564444
SAR = 3FE ( 63) VOUT = 3.280000
SAR = 1FE ( 31) VOUT = 3.137778
SAR = FE ( 15) VOUT = 3.066667
SAR = 7E ( 7) VOUT = 3.031111
SAR = 3E ( 3) VOUT = 3.013333
SAR = 1E ( 1) VOUT = 3.004444
SAR = 0E ( 0) VOUT = 3.000000
SAR = 1E ( 1) which means 0.005000 volts
```



```

#include "CTYPE.H"
#include "STDIO.H"
/* example invocation lines

a2dsim 0.0025 3.0 0.07 p
Vin Vbias Voffs print to screen and lp

a2dsim 0.0075 3.0 0.07
Vin Vbias Voffs print to screen only the first trial input

*/
int main(k, argv)
int k;
char *argv[];
{
    FILE *fp, *fopen();
    double initial_conditions, vin, vout, vcent, voff, v89, v19;
    unsigned int sar = 0x3FF;
    unsigned int mask = 0x200;
    unsigned int count = 0;
    unsigned int printon;
    if (strcmp(argv[0], "run") == 0)
        count++;
    if ((k != (4 + count)) & (k != (5 + count)))
    {
        printf("\nInvocation error!\n");
        return;
    }
    count++;
    sscanf(argv[count++], "%lf", &vin);
    sscanf(argv[count++], "%lf", &vcent);
    sscanf(argv[count++], "%lf", &voff);
    if (count == k)
        printon = 0;
    else printon = 1;
    printf("A to D simulator.(center taps)..");
    if (printon)
    {
        if ((fp = fopen("prn:", "w")) == 0)
        {
            printf("\nCan't open printer\n");
            return;
        }
    }
    if (printon)
        fprintf(fp, "A to D simulator..");
    printf(" with \nVin = %f\nVcent = %f\nVoff = %f\n", vin, vcent, voff);
    if (printon)
        fprintf(fp, " with \nVin = %f\nVcent = %f\nVoff = %f\n",
            vin, vcent, voff);

    initial_conditions = ((8.0 / 9.0) * (vcent - vin))
        + ((1.0 / 9.0) * (vcent - voff));
    v89 = 8.0 / 9.0;
    v19 = 1.0 / 9.0;
}

```

Listing B1. A/D Converter Simulator

```

sar ^= mask;
printf("SAR = %3xH (%4d)\t", sar, sar);
if (printon)
    fprintf(fp, "SAR = %3xH (%4d)\t", sar, sar);
for (count = 0; count < 10; count++)
{
    vout = (v89 * ((double)(sar >> 2)) * 0.02 + 0.10)
    + (v19 * (((double)(sar & 3)) * 0.02 + 0.01))
    + initial_conditions;
    if (vout > vcent)
        sar |= mask;
        mask <<= 1;
        sar ^= mask;
        printf("Vout = %f\nSAR = %3xH (%4d)\t", vout, sar, sar);
        if (printon)
            fprintf(fp, "Vout = %f\nSAR = %3xH (%4d)\t", vout, sar, sar);
        vout, sar, sar);
    printf("which means %f volts\n", ((double)sar * 0.005));
    if (printon)
        fprintf(fp, "which means %f volts\n", ((double)sar * 0.005));
    return;
}
/* main */

```

270365-A6

Listing B1. A/D Converter Simulator (Continued)

APPENDIX E BIBLIOGRAPHY

- A/D Processing with Microcontrollers, Katausky, IEEE STD. 746-1984
Horden, Smith
- Apfel, R., et. al., "Signal-Processing Chips enrich telephone line- card Architecture". Electronics, May 5, 1982.
- Analog Devices - Data-Acquisition Databook 1984, Volume 1
- Blahut, Richard E., "Fast algorithms for digital signal processing", Addison Wesley Publishing Company, Inc., 1985.
- Boyes, ed. - Syncro and Resolver Conversion, 1980
- Brown, Robert Grover, "Introduction to random signal analysis and Kalman filtering". John Wiley & Sons, Inc., 1983.
- Burr-Brown Application Note, Testing of Analog-to-Digital Converters
- Burton and Dexter - Microprocessor Systems Handbook, 1977
- Candy, J., et. al., "The Structure of Quantization Noise from Sigma-Delta Modulation", IEEE Transaction on Comm. Vol. Com. 29, No. 9, Sept. 1981.
- Candy, J., et. al., "Using Triangularly Weighted Interpolation to Get 13-Bit PCM from a Sigma-Delta Modulator", IEEE Transaction on Comm., Nov. 1976.
- Electronic Analog-to-Digital Converters, Seitzer, Pretzl, Handy
- Handbook of Electronic Calculations, Chapter 15, Analog-Digital Conversion
- Harris Analog and Telecom Data Book
- IEEE 162
- Intel Application Note AP-124 - High-Speed Digital Servos for Motor Control Using the 2920/21 Signal Processor
- Intel Application Note AP-125 - Designing Microcontroller Systems for Electrically Noisy Environments
- Irwensen, J., "Calculated Quantization Noise of Single Integration Delta Modulation Coders" BSTJ Sept. 1969.
- ITT Digital 2000 VLSI Digital TV System, MAA 2300 Audio A/D Converter, Edition 1983/9.
- MIL-M-38510/135 June 4, 1984
- MIL-M-38510/135 May 6, 1985
- Modern Electronic Circuits Reference Manual
- NBS Staff Reports, May/June 1981 P.22/23
- Sheingold, ed. - Analog-Digital Conversion Handbook, 1972
- Sheingold, ed. - Analog-Digital Conversion Notes, 1977
- Sheingold, ed. - Non-Linear Circuits Handbook, 1974
- Sheingold, ed. - Transducer Interfacing Handbook, 1980
- Steele, R., "Delta Modulation Systems", Pentech Press Limited, 1975.
- Taylor, Fred U., "Digital filter design handbook", Marcel Dekker, Inc., 1983.
- Terminology Related to the Perf of S/H, A/D, D/A Circuits, IEEE Transactions

February 1984

Order Number: 231040-001

High Performance Event Interface For A Microcomputer

As silicon technology advances to provide denser geometries, timer structures have become more elegant and powerful.

by Lionel Smith, Intel Corp.

Microcontrollers are microprocessors specially configured to monitor and control mechanisms and processes rather than manipulate data. The systems they are imbedded in are often called real time control systems; microcontrollers always incorporate some form of timer structure to allow synchronization with the outside or 'real' world. As silicon technology advances to provide denser geometries, these structures have become more elegant and powerful.

This trend can be seen in the Intel 8048, the Motorola 6801, and the Intel 8051 which were introduced at approximately two and a half year intervals starting in 1976. The 8048 has a single 8-bit timer; the 6801 has a 16-bit timer, and the 8051 has two 16-bit timers. The new 16-bit microcontroller from Intel, the 8096, has an independent High Speed I/O subsystem which provides the functionality of four to eight 16-bit timers. While this subsystem is designed to provide an integrated approach to measuring and controlling time modulated signals, it is easier to describe as separate input and output units.

Lionel Smith is a Staff Applications Engineer in the Microcontroller Operations Division of Intel Corporation, Chandler, Arizona. During his eight years with Intel Corp., Mr. Smith participated in the definition of the Intel 8051 and 8096 microcontroller architectures. He is also the author of several application notes on microcontrollers.

High Speed Input Unit

The purpose of the High Speed Input unit is to allow the measurement of the periods of incoming pulse or frequency modulated inputs with high resolution and minimal software overhead. A block diagram of the hardware used to accomplish this goal is shown in Figure 1. The heart of this unit is a programmable change detector which monitors the four I/O pins of the 8096 which are designated as "High Speed Inputs" (HSI.0-HSI.3).

The operating mode of the change detector is controlled by a byte register which can be written as register 3 of the onboard register file. This register has the predeclared name HSLMODE in the 8096 assembly language. The register contains a separate field for each of

the four HSI pins. There are two bits in each of these fields and they are encoded as follows:

- 00 Capture every eighth positive transition
- 01 Capture positive transitions only
- 10 Capture negative transitions only
- 11 Capture both positive and negative transitions

It is also possible to disconnect one or more of the HSI pins from the change detector by writing into one of the two I/O control registers. This register, known to the assembly language as IOCO is addressed as register 15H of the on-board register file. HSI pins that

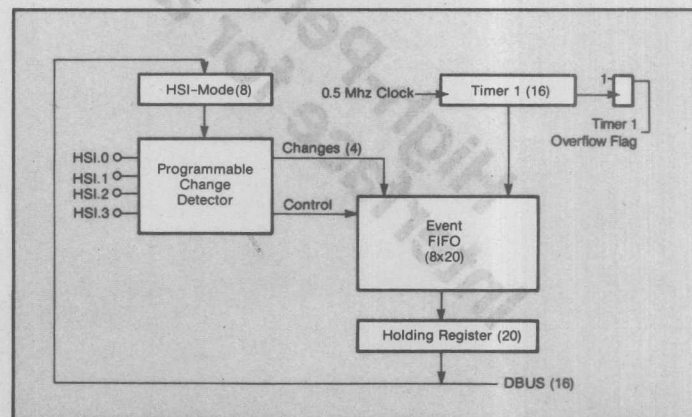


Figure 1: Diagram shows the High-Speed Input Unit which is used to measure incoming pulse or frequency modulated inputs.

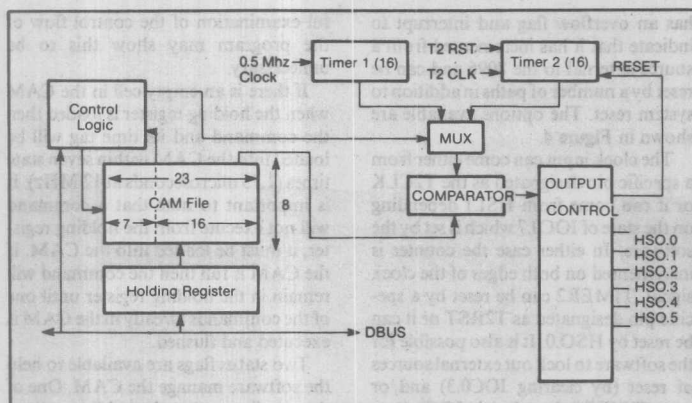


Figure 2: Block diagram of the High Speed Output hardware.

have to be disconnected from the change detector are available for use as normal digital inputs and two of them (HSI.2 and HSI.3) can be used by the High Speed Output unit.

When a change (or changes) of the required type occurs, four bits of change information, along with the current value of TIMER1, are loaded into a FIFO (first-in, first-out memory). Each set bit in this field indicates that a change occurred on the corresponding input pin.

The time reference for the HSI unit is TIMER1, a sixteen bit counter which is incremented every eight state times by the CPU clock. With a 12 MHz crystal this gives a resolution of 2.0 microsec-

onds. TIMER1 is cleared by reset and then starts incrementing. It cannot be written to by the software but can be read as a sixteen bit word at any time. When its count goes from all ones to zero a flag is set and an interrupt generated. The software can use this flag and/or interrupt to extend the measurement range of the HSI unit.

The FIFO that is used to store the change and time information is eight levels deep (including the holding register) and 20 bits wide. The oldest entry in the FIFO is placed in the holding register. When the holding register is read then the next oldest entry will drop into it and another cell of the FIFO will become available for input data. An

interrupt can be generated either when one or more entries exist in the FIFO or when seven or more entries exist. The choice is made by the software by setting a bit in I/O control register 1 (IOC1).

The 8096 only supports byte and word operands for most operations. The holding register is 20 bits wide hence the holding register is broken down into two registers. The 16-bit time field is read as a word register and is known as HSLTIME to the assembler. The change information is read as an eight-bit byte known as HSLSTATUS. The four extra bits in this byte are used to report the state of the HSI pins at the time the register is read (not at the time the reported change occurred). The holding register is cleared after the HSLTIME is read so that HSLSTATUS can be read at any time to monitor the actual state of the HSI pins without losing data from the FIFO.

High Speed Output Unit

The High Speed Output unit serves the output requirements of the system in the same way as the HSI unit serves the input. It allows the generation of pulse and frequency modulated signals with high resolution and minimal software overhead. It can also be used to generate time delays for the operating software and to trigger the A/D converter at precise time intervals for signal processing algorithms. A block diagram of the HSO hardware is shown in Figure 2.

The HSO unit is driven by a Content Addressable Memory (CAM) which is 23 bits wide and eight levels deep. The

(continued on page 120)

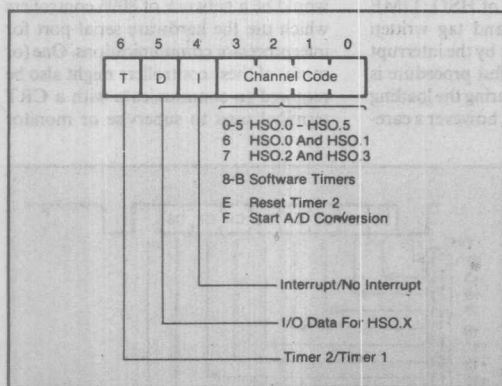


Figure 3: Diagram shows format of Command Tag. The lower four bits specify the basic operation and the remaining three bits are options to the basic operation.

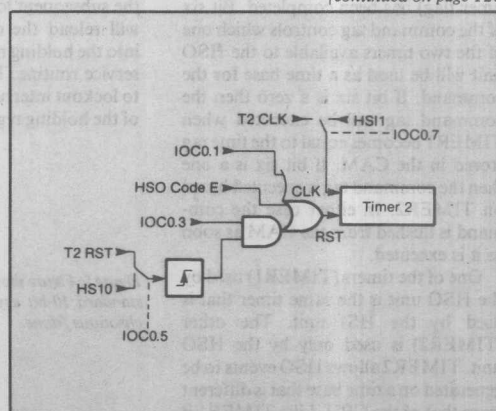


Figure 4: Figure shows the clock and reset options of TIMER2 used by the High Speed Output unit.

(continued from page 112)

23 bits are broken into a 16 bit time tag and a seven bit command tag. The command tag tells it when to do it. The format of the command tag is shown in Figure 3. The lower four bits of the tag specify the basic operation and the remaining three bits specify options to the basic operation. The basic operations supported are:

- Write to one of the six pins controlled by the HSO unit (HSO.0-HSO.5).
- Write to HSO.0 and HSO.1 with a single command.
- Write to HSO.2 and HSO.3 with a single command.
- Set one of four software timer flags.
- Reset Timer 2.
- Trigger an A/D conversion.

If an operation on an HSO pin is specified, then the value to be written to the pin is taken from bit five of the command tag. Note that if two HSO pins are to be modified with the same command then both will be set to the same state. Bit five of the command tag is ignored for the other HSO operations. Bit four of the command tag enables the generation of an interrupt which occurs when the command is executed.

There are two interrupts generated by the HSO unit. One of them indicates that an operation involving a HSO pin has occurred, and the other is used to signal that one of the internal HSO functions (such as setting a software timer flag) has been completed. Bit six of the command tag controls which one of the two timers available to the HSO unit will be used as a time base for the command. If bit six is a zero then the command tag will be executed when TIMER1 becomes equal to the time tag stored in the CAM. If bit six is a one then the command tag is executed based on TIMER2. In either case the command is flushed from the CAM as soon as it is executed.

One of the timers (TIMER1) used by the HSO unit is the same timer that is used by the HSI unit. The other (TIMER2) is used only by the HSO unit. TIMER2 allows HSO events to be generated on a time base that is different from that of the CPU. Like TIMER1 it is a 16 bit counter that can be read but not written to by the software. It also

has an overflow flag and interrupt to indicate that it has incremented from a source external to the 8096 and can be reset by a number of paths in addition to system reset. The options available are shown in Figure 4.

The clock input can come either from a specific pin designated as the T2CLK or it can come from HSI.1 depending on the state of IOC0.7 which is set by the software. In either case the counter is incremented on both edges of the clock signal. TIMER2 can be reset by a specific pin designated as T2RST or it can be reset by HSO.0. It is also possible for the software to lock out external sources of reset (by clearing IOC0.3) and/or reset TIMER2 directly (via IOC0.1) or indirectly via a command stored in the CAM. Note that this last possibility allows TIMER2 to be configured as a modulation counter since the software can command the HSO unit to clear TIMER2 when it reaches a given value.

Commands are loaded into the CAM from the 23 bit wide holding register which, like the holding register for the HSI unit, is actually made up of a byte register (HSO_COMMAND) which stores the command tag and a word register is considered loaded after HSO_TIME is loaded so the software must always load HSO_COMMAND and then load HSO_TIME.

The software must also ensure that the loading of the two registers is not interrupted by an interrupt service routine which uses the HSO unit. If such an interrupt occurs immediately following the loading of HSO_COMMAND then the subsequent loading of HSO_TIME will reload the command tag written into the holding register by the interrupt service routine. The safest procedure is to lockout interrupts during the loading of the holding registers, however a care-

ful examination of the control flow of the program may show this to be unnecessary.

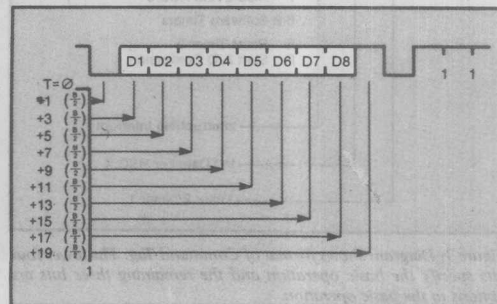
If there is an empty cell in the CAM when the holding register is loaded then the command and its time tag will be loaded into the CAM within seven state times (1.75 microseconds at 12 MHz). It is important to note that a command will not execute from the holding register, it must be loaded into the CAM. If the CAM is full then the command will remain in the holding register until one of the commands already in the CAM is executed and flushed.

Two status flags are available to help the software manage the CAM. One of them indicates that the holding register is full or the CAM is full. Once a command is loaded into the CAM it cannot be read or overwritten, it can only be flushed after it is executed. To support those situations where the software wishes to cancel a command after it has been loaded, the HSO unit is configured so that two operations to a HSO pin which cancel each other will not effect the setting of the pin if they are executed with identical time tags.

Application Example

Since the 8096 incorporates a full duplex asynchronous serial port in its hardware it may seem strange that one would want to implement a software driven serial port using the high speed I/O features. There are, however, many useful configurations of microcontroller systems which in fact require more than a single serial port. An obvious example would be a network of 8096 controllers which use the hardware serial port for interprocessor communications. One (or more) of these controllers might also be required to communicate with a CRT terminal used to supervise or monitor

Figure 5: Figure shows standard 10-bit asynchronous frame.



The serial output process is simpler than the receive process because there is no need to synchronize with the outside world.

the system. Another example would be a simple CRT terminal design based on an 8096 which needs one serial port for communication and another for driving a slave printer. It may also be true that this is, in fact, a strange requirement. In any case it is an excellent example to show how the high speed I/O features of the 8096 might be used.

The objective is to add a software driven asynchronous serial port to the 8096 that provides full duplex serial communication at 2400 baud. A standard frame consisting of a START bit, eight data bits and a STOP bit will be assumed. A high speed input pin (HSI.0) will be used for received data and a high speed output pin (HSI.0) for transmit data.

A standard 10-bit asynchronous frame is shown in Figure 5. The figure also shows the points in time where the receive process must sample the incom-

ing data stream and take some action. The first timing point (labeled T=0) is the leading edge of the start bit, the accurate sensing of this edge is important because all subsequent sample times are relative to this edge. This event also places the highest burden on the sampling algorithm because it can occur at any point in time. The rest of the sampling events occur at some multiple of one-half a bit period relative to the edge of the start bit. The diagram uses the symbol B to represent a bit period.

At the second sample, which occurs half way through the start bit, the data must be checked to make sure it is still a SPACE. If it is not, a noise pulse has caused a false start and the receive process must be reinitialized. The next eight samples are used to shift in the serial data stream. The last sample, which occurs 19 one-half bit times after the leading edge of the start bit, is used to verify that the stop bit is valid (i.e. it is in the MARK state). If it is not the a framing error must be reported since it is likely that the receiver is not properly synchronized with the transmitter.

The HSI unit is an ideal mechanism for detecting the leading edge of the start bit. All that needs to be done is to set the mode register to detect negative going edges on HSI.0.

The software timer interrupt service routine implements a simple state machine based on the variable count. The routine also arranges for the next sample by issuing a command to the HSO unit to generate another software

timer interrupt at the appropriate time. This is done in all states unless the reception of the character is complete or a false START bit has been detected. Under these conditions the receive process must be reinitialized by enabling HSI.0 into the event FIFO (by setting IOC0.0) instead of retriggering the software timer.

The serial output process is simpler than the receive process because there is no need to synchronize with the outside world. A transmission can be started at any time by setting the TxD line to a space for one bit time to form the START bit. Following the START bit are the eight data bits and the STOP bit. The HSO interrupt service routine can be used to transmit the data and the stop bit but the transmit process must be initialized.

The only real complication in the HSO interrupt service routine is that there are no flags available in the 8096 which indicate which of the HSO outputs caused the interrupt. In many systems this does not represent a problem because the HSO unit can be treated as a write only device. It is given commands which are to be executed at the proper time but no feedback is required to indicate when the proper time has been reached. In this case, however, the feedback is required since the CAM isn't deep enough to hold all of the transitions required for a character. Even if it were big enough it is unlikely that so many CAM locations would be dedicated to serial output. □

These interrupt at the appropriate time. This is done in all states within the logic. If the character is complete or a later START bit has been detected. Under these conditions the receive process must be reinitialized by enabling H210 into the event FIFO (by setting I0C0D) instead of retriggering the software timer.

The serial output process is simpler than the receive process because there is no need to synchronize with the outside world. A transmission can be started at any time by setting the TXD line to a high level.

Following the START bit, the eight data bits and the STOP bit. The H20 interrupt service routine can be used to transmit the data and the stop bit but the transmit process must be reinitialized.

The only real complication in the H20 interrupt service routine is that there are no flags available in the 8086 which indicate which of the H20 outputs caused the interrupt. In many systems this does not represent a problem because the H20 unit can be treated as a write only device. It is given commands which are to be executed at the proper time but no feedback is required to indicate when the proper time has been reached. In this case, however, the feedback is required since the CAM unit is deep enough to hold all of the data. It is required for a character. Even if it were big enough it is unlikely that so many CAM locations would be dedicated to serial output.

ing data stream and take some action. The first timing point (labeled T=0) is the leading edge of the start bit. The accuracy of this edge is important because all subsequent sample times are relative to this edge. This event also places the highest burden on the sampling algorithm because it can occur at any point in time. The rest of the sampling events occur at some multiple of one-half a bit period relative to the edge of the start bit. The diagram uses the symbol B to represent a bit period.

At the second sample, which occurs half a bit period after the start bit, the data must be checked to make sure it is still a SPACE. If it is not, a noise pulse has caused a false start and the receive process must be reinitialized. The next eight samples are used to shift in the serial data stream. The last sample, which occurs 19 one-half bit times after the leading edge of the start bit, is used to verify that the stop bit is valid (i.e. it is in the MARK state). If it is not the a framing error must be reported since it is likely that the receiver is not properly locked with the transmitter.

The H21 unit is an ideal mechanism for the leading edge of the start bit. It is to be done is to set the H21 output to a high level. The H21 output is a high speed input pin (H210) will be used for received data and a high speed output pin (H210) for transmit data.

The serial output process is simpler than the receive process because there is no need to synchronize with the outside world.

January 1985

Motor Controllers Take the Single-Chip Route

Ira Horden, John Katausky
and Lionel Smith
Intel Corporation
Chandler, AZ

Another example would be a simple CRT terminal design based on an 8086 which needs one serial port for communication and another for driving a slave printer. It may also be true that this is in fact a stringent requirement. In any case it is an excellent example to show how the high speed I/O features of the 8086 might be used.

The objective is to add a software driven asynchronous serial port to the 8086 that provides full duplex serial communication at 2400 baud. A standard frame consisting of a START bit, eight data bits and a STOP bit will be assumed. A high speed input pin (H210) will be used for received data and a high speed output pin (H210) for transmit data.

A standard 10-bit asynchronous frame is shown in Figure 2. The figure also shows the points in time where the receive process must sample the incoming data stream and take some action.

Motor controllers take the single-chip route

Adaptable to a wide range of applications, a microcontroller can lower cost and boost reliability, compared to multichip units

Ira Horden, John Katausky,
and Lionel Smith
Intel Corp., Chandler, AZ

In the quest for accurate positioning of motors in automated industrial machinery or other controlled environments, the single-chip motor controller is gaining ground. Most of the newer single-chip controllers use an internal microprocessor combined with analog and digital circuitry optimized for motion control. In addition to being more flexible and reducing IC count, such a controller often means lower cost and higher reliability.

A microcontroller provides a high level of functionality by combining the microprocessor, RAM, ROM, and I/O ports, with other functions associated with control and feedback. What's more, the instruction set of a microcontroller is usually optimized for fast interrupt response and processing, in contrast to general-purpose microprocessors that are structured for data processing.

In essence, most control systems are similar (see Fig. 1), although each has its own application-dependent transducers to accurately monitor the process under control. The system blocks will include:

- A position and velocity decoder that determines the position, velocity, or both of the controlled motor, based on information fed back from the external sensing hardware. In a high-performance system, the CPU should be able to access this information directly.

- A control-panel decoder, which reads the settings of the control panel and translates this data to a format useable by the CPU. This section may include a serial data link if the control panel is remotely located or if inter/intracomputer communications is required.

- An error calculator, which compares desired values of position, velocity, or any closed-loop data being monitored and controlled to the desired position, velocity, and so on.

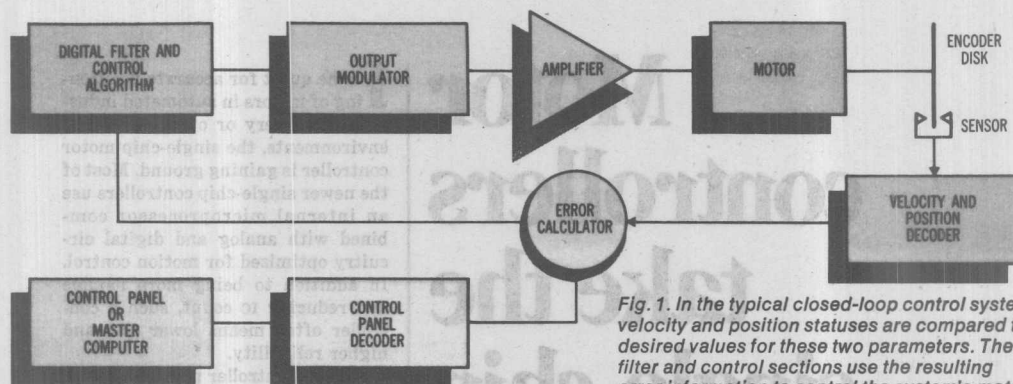


Fig. 1. In the typical closed-loop control systems, velocity and position statuses are compared to desired values for these two parameters. The filter and control sections use the resulting error information to control the system's motor.

The output is presented to a filtering and control stage (usually digital), which uses this data to implement the control algorithm.

- A filtering and control-algorithm executor, which executes the control algorithm using data obtained from previous blocks. The resulting response is the signal that eventually is used to control and adjust the motor.
- An output modulator, which adjusts the output to the motor, based on the information presented by the filter and control-algorithm executor.

Algorithms control

Many different algorithms are used for control purposes. Some may require many arithmetic multiplications and divisions, which means that the hardware controller should supply these capabilities quickly enough to respond accurately. But since most controllers have reasonably fast execution times, the control algorithm may be selected to match the controller after the microcontroller has been selected.

Combining a microcontroller like the Intel 8096 with a few level-shifting devices, creates a simple control system (see Fig. 2). By properly selecting algorithms for the hardware, the system could control one motor to a high degree of speed and

accuracy or several motors with reasonable speed and accuracy. Configurable I/O features and fast execution times make the 8096 and other state of the art controllers, the best vehicles for solving a variety of control problems.

The architecture of microcontrollers also provides advantages over general-purpose microprocessors; position and velocity decoding, control panel decoding, and output encoding are simplified. Although the type of encoding and decoding depends on the characteristics of the system, the microcontroller should optimize the scheme employed.

For example, a common system configuration uses a quadrature encoder for position and velocity feedback, a serial data link to the control panel or central computer, and a linear amplifier to drive the motor. A quadrature encoder is used because it provides information on direction as well as speed of rotation (see Fig. 3). The microcontroller derives the direction from the phase relationship of the pulse edges. By gating and counting pulses, it extracts the speed of rotation.

However, the frequency of the encoder is limited by the ability of the decoder to count and respond to pulses. If the processor is not fast enough, external counters under

control of the processor must be used. Here too, the complexity of the algorithm chosen affects the performance of the controller needed.

A minimum encoder period of 100 μ s or greater allows most processors to execute a simple algorithm. As the algorithm increases in complexity, more cycles are required; thus less time is available between encoder pulses to implement the algorithm. Although many clever algorithms and coding practices allow high levels of performance, the inherent power of the controller in charge can be the deciding factor between success and failure.

Controller has many features

The 8096 16-bit microcontroller from Intel, contains many of the high-level features needed to allow its use in many applications (see sidebar "Inside the 8096"). It is configured easily in a variety of ways, allowing it to tackle the various problems in different ways. For example, there are two ways its high-speed input/output unit (HSIO) can be used to monitor quadrature encoding. One way uses an external counter, freeing one of the microcontroller's two internal counters for other functions.

If only one motor is to be controlled and the two internal counters

are used, timer 2 counts the input pulses. This allows encoder periods as short as 4 μ s. When programmed to do so, the high-speed output unit (HSO) will interrupt when the timer reaches a certain value. This is useful if a large number of en-

coder pulses will be received between changes of motor speed.

The two timers play a major role when a motor is to be accelerated from a stop condition. The HSO interrupts the CPU when the motor reaches approximately 75% of its

maximum speed, thereby reducing the voltage to the motor for smooth and controlled acceleration to full speed (see Fig. 4).

The initial application of high voltage means high starting current (thus torque). By changing voltage

Inside the 8096

The 8096, Intel's most recently introduced 16-bit microcontroller, went into volume production last month. It combines a large number of loosely coupled features, yielding a high flexible controller. The CPU at the heart of the 8096 microcontroller includes a register file, a register ALU (RALU), and a control unit.

The control unit contains logic arrays which decode RALU instructions. The RALU is able to operate on any byte or word in the register file. This eliminates accumulator bottlenecking, which occurs often in standard accumulator dominated architectures.

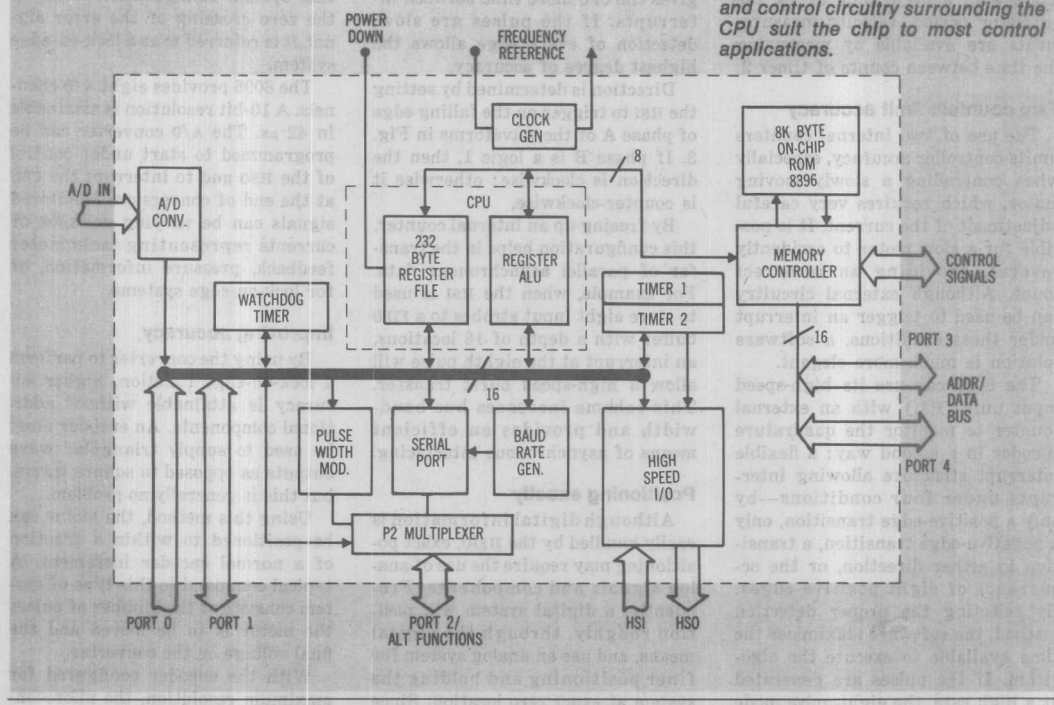
Additionally, algorithms execute quicker because the design eliminates much of the data transferral necessary when a bottleneck architecture is employed.

Special I/O functions are accessed through special function registers, which are certain reserved registers within the 256 bytes of RAM making up the register file. Special functions within the 8096 include an A/D converter, a pulse-width-modulator (PWM) output, a multimode serial port, a watchdog timer, and a unique high-speed input/output (HSIO) unit.

The HSIO consists of three

parts, the high-speed input (HSI) unit, the high-speed output (HSO) unit, and two 16-bit timers (one of which can be operated as a counter). The HSO's outputs and interrupts can be synchronized to either of the timers. The start conversion command to the A/D converter can be triggered under control of the HSIO, as can recording the value of the timer when an input took place. By using the HSIO to unburden the CPU, I/O monitoring routines are unnecessary and throughput is increased.

The 8096's CPU consists of a 256-byte register file, an RALU (register ALU), and the control unit. Support and control circuitry surrounding the CPU suit the chip to most control applications.



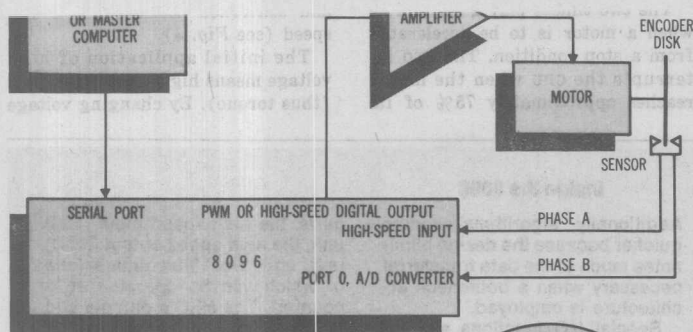


Fig. 3. An optical chopper disk used as a quadrature encoder provides information on both direction and speed of rotation. The phase relationships of the waveforms determine direction, and their frequency determines velocity.

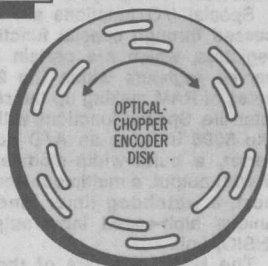
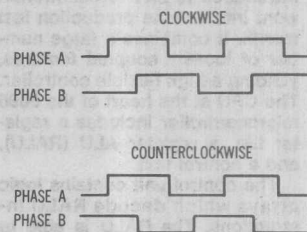


Fig. 2. The 8096 functions well as the heart of a motor-control system. Needing few support circuits, it can control one motor with a high degree of accuracy or several motors with reasonably good accuracy.



INSIDE TRACK GENERATES PHASE A
OUTSIDE TRACK GENERATES PHASE B

at the appropriate time, the controller provides a smooth acceleration. If timer 1 is used in a free-running mode with a cycle time of $\frac{1}{4}$ the oscillator period, velocity measurements are available by measuring the time between counts of timer 2.

Two counters limit accuracy

The use of two internal counters limits controller accuracy, especially when controlling a slowly moving motor, which requires very careful adjustment of the current. It is possible for a slow motor to accidentally reverse, providing an incorrect count. Although external circuitry can be used to trigger an interrupt under these conditions, a software solution is much more elegant.

The 8096 can use its high-speed input unit (HSI) with an external counter to monitor the quadrature encoder in a second way: a flexible interrupt structure allowing interrupts under four conditions—by only a positive-edge transition, only a negative-edge transition, a transition in either direction, or the occurrence of eight positive edges. By selecting the proper detection method, the software maximizes the time available to execute the algorithm. If the pulses are generated at a high rate, the eight-pulse mode

gives the CPU more time between interrupts. If the pulses are slow, detection of every edge allows the highest degree of accuracy.

Direction is determined by setting the HSI to trigger on the falling edge of phase A of the waveforms in Fig. 3. If phase B is a logic 1, then the direction is clockwise; otherwise it is counter-clockwise.

By freeing up an internal counter, this configuration helps in the transfer of parallel asynchronous data. For example, when the HSI is used to store eight input strobes to a FIFO buffer with a depth of 16 locations, an interrupt at the eighth pulse will allow a high-speed burst transfer. This scheme increases bus bandwidth and provides an efficient means of asynchronous interfacing.

Positioning exactly

Although digital information is easily handled by the HSIO, exact positioning may require the use of analog signals and components. Frequently, a digital system will position roughly, through the digital means, and use an analog system for finer positioning and holding the system at exact zero location. Since

this system configuration locks on the zero crossing of the error signal, it is referred to as a lock-on-edge system.

The 8096 provides eight A/D channels. A 10-bit resolution is attainable in 42 μ s. The A/D converter can be programmed to start under control of the HSO and to interrupt the CPU at the end of conversion. Monitored signals can be varying voltages or currents representing tachometer feedback, pressure information, or for lock-on-edge systems.

Improving accuracy

By using the converter to perform a lock-on-edge function, higher accuracy is attainable without additional components. An encoder must be used to supply triangular wave outputs as opposed to square waves, but this is generally no problem.

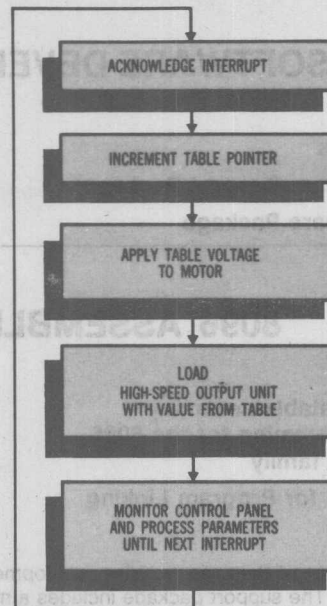
Using this method, the motor can be positioned to within a fraction of a normal encoder increment. A typical command to this type of system consists of the number of pulses the motor is to be moved and the final voltage on the converter.

With the encoder configured for maximum resolution, the microcon-

troller can implement the desired algorithm. To effectively drive the motor, it uses a pulse-width-modulation scheme that generates an analog output. An 8-bit timer controls the PWM output by turning the output on when the timer overflows and turning it off when the timer matches the contents of an internal PWM register.

Since incrementing of the timer occurs every clock cycle, a 12-MHz clock input results in a 64- μ s period of the PWM waveform. It is necessary to buffer and amplify the PWM output, and depending on the motor and amplifier, an RC filter may be used to smooth out the output.

Another means of generating analog voltages uses the HSO to toggle at the desired times. The disadvantage of this technique is only eight operations can be pending in the HSO at any one time. This means that the CPU is more loaded down than when the PWM section is used. Although CPU time may suffer, this method allows up to six analog outputs to be generated. This technique may be necessary if the PWM register is already used for another function. □



Voltage and Pulse-Count Table		
% of total pulses	Current count of pulses	% of full voltage
75%	7,500	100%
85%	8,500	50%
90%	9,000	25%
95%	9,500	10%
97%	9,700	04%
100%	10,000	00%

Fig. 4. The iterative process of control loops around while monitoring change requests from the control panel. Varying the applied voltage to the motor in a predetermined way, as shown in the table, smooths acceleration and deceleration.

Reprinted From **ELECTRONIC PRODUCTS Magazine** — January 1, 1985
645 Stewart Ave., Garden City, NY 11530 • © 1985 Hearst Business Communications Inc. PRINTED IN U.S.A.



8096 SOFTWARE DEVELOPMENT PACKAGES

- Choice of Hosts
- MCS®-96 Software Support Package
- C-96/196 Software Package
- Supports All Members of the MCS-96 Family
- PL/M-96 Software Package

8096 ASSEMBLER PACKAGE

- Symbolic relocatable assembly language programming for the 8096 microcontroller family
- System Utilities for Program Linking and Relocation
- Extends Intellec® Microcomputer Development System to support 8096 program development
- Encourages modular program design for maintainability and reliability

The 8096 Software Support Package provides development system support for the 8096 family of 16-bit single chip microcomputers. The support package includes a macro assembler and system utilities.

The assembler produces relocatable object modules from 8096 macro assembly language instructions. The object modules then are linked and located to absolute memory locations.

The assembler and utilities run on PC DOS 3.0 IBM® PC XT/AT Systems.

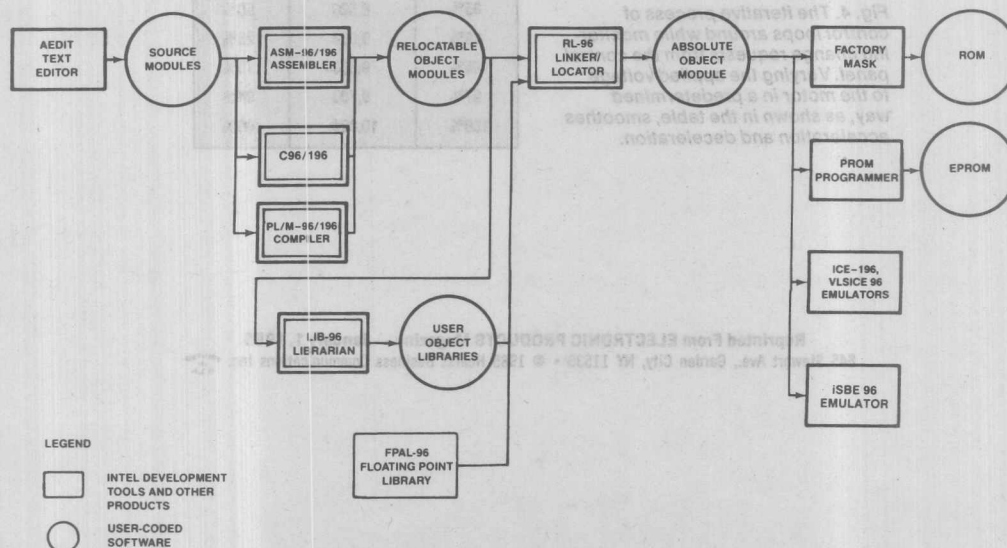


Figure 1. 8096 Software Development Process

230613-1

*IBM is a registered trademark of International Business Machines Corporation.

8096 MACRO ASSEMBLER

- Gives Symbolic Access to Powerful 8096 Hardware Features
- Symbolic Assembler Supports Macro Capabilities, Cross Reference, Symbol Table and Conditional Assembly
- Object Files are Linkable and Locatable

ASM-96 is the macro assembler for the MCS family of microcontrollers, including the 80C196. ASM-96 translates symbolic assembly language mnemonics into relocatable object code. Since the object modules are linkable and locatable, ASM-96 encourages modular programming practices.

The macro facility in ASM-96 allows programmers to save development and maintenance time since common code sequences only have to be done once. The assembler also provides conditional assembly capabilities.

ASM-96 supports symbolic access to the many features of the 8096 architecture. An "include" file is provided with all of the 8096 hardware registers defined. Alternatively, the user can define any subset of the 8096 hardware register set.

Math routines are supported with mnemonics for 16×16 -bit multiply or $32/16$ -bit divide instructions.

The assembler runs on a PC-DOS 3.0 IBM PC XT/AT.

RL96 LINKER AND RELOCATOR PROGRAM

- Links Modules Generated by ASM-96, C-96, and PL/M-96
- Encourages Modular Programming for Faster Program Development
- Locates the Linked Object Module to Absolute Memory Locations
- Automated Selection of Required Modules from Libraries to Satisfy Symbolic References

RL96 is a utility that performs two functions useful in MCS-96 software development:

- The link function which combines a number of MCS-96 object modules into a single program.
- The locate functions which assigns an absolute address to all relocatable addresses in the MCS-96 object module.

RL96 resolves all external symbol references between modules and will select object modules from library files if necessary.

RL96 creates two files:

- The program or absolute object module file that can be executed by the targeted member of the MCS-96 family.
- The listing file that shows the results of link/locate, including a memory map symbol table and an optional cross reference listing.

The relocater allows programmers to concentrate on software functionally and not worry about the absolute addresses of the object code. RL96 promotes modular programming. The application can be broken down into separate modules that are easier to design, test and maintain. Standard modules can be developed and used in different applications thus saving software development time.

FPAL96 FLOATING POINT ARITHMETIC LIBRARY

- Implements IEEE Floating Point Arithmetic
- Basic Arithmetic Operations
+, -, ×, /, Mod Plus Square Root
- Supports Single Precision 32 Bit Floating Point Variables
- Includes an Error Handler Library

FPAL96 is a library of single precision 32-bit floating point arithmetic functions. All math adheres to the proposed IEEE floating point standard for accuracy and reliability. An error handler to handle exceptions (for example, divide by zero) is included.

The following functions are included:

ADD	NEGATE
SUBTRACT	ABSOLUTE
MULTIPLY	SQUARE ROOT
DIVIDE	INTEGER
COMPARE	REMAINDER

LIB 96

The LIB 96 utility creates and maintains libraries of software object modules. The customer can develop standard modules and place them in libraries. Application programs can then call these modules using predefined interfaces.

LIB 96 uses the following set of commands:

- CREATE: Creates an empty library file.
- ADD: Adds object modules to a library file.
- DELETE: Deletes object modules from a library file.
- LIST: Lists the modules in the library file.
- EXIT: Terminates LIB 96

When using object libraries, RL96 will include only those object modules that are required to satisfy external references, thus saving memory space.

ORDERING INFORMATION

Order Code	Operating Environment
D86ASM96	96 Assembler for PC DOS 3.0 Systems

Documentation Package:

MCS-96 Macro Assembler User's Guide
MCS-96 Utilities User's Guide
MCS-96 Assembler and Utilities Pocket Reference Card
8096 Floating Point Arithmetic Library

SUPPORT:

Hotline Telephone Support, Software Performance Report (SPR), Software Updates, Technical Reports, and Monthly Technical Newsletters are available.

PL/M-96 SOFTWARE PACKAGE

- Choice of Hosts
- Block Structured Language Design Encourages Module Programming
- Provides Access to 8096 on Chip Resources
- Produces Relocatable Object Code which is Linkable to Object Modules Generated by Other 8096 Translators
- Resident on 8086 Intel Microcomputer Development Systems for Higher Performance
- Includes a Linking and Relocating Utility and the Library Manager
- IEEE Floating Point Library included for Numeric Support
- Compatible with PL/M-86 Assuring Design Portability

PL/M-96 is a structured, high-level programming language useful for developing software for the Intel 8096 family of microcontrollers, including the 80C196. PL/M-96 was designed to support the software requirements of advanced 16 bit microcontrollers. Access to the on chip resources of the 8096 has been provided in PL/M-96.

PL/M-96 is compatible with PL/M-86. Programmers familiar with PL/M will find they can program in PL/M-96 with little relearning effort.

The PL/M-96 compiler translates PL/M-96 high level language statements into 8096 machine instructions. By programming in PL/M an engineer can be more productive in the initial software development cycle of the project. PL/M can also reduce future maintenance and support cost because PL/M programs are easier to understand. PL/M-96 was designed to complement Intel's ASM-96.

PL/M-96 is available for PC DOS 3.0 based IBM PC XT/AT Systems.

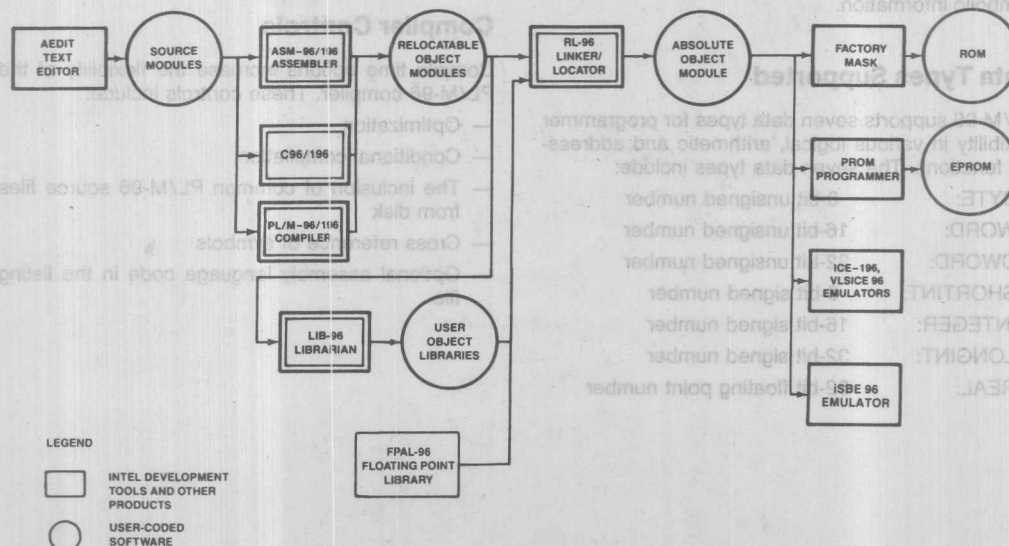


Figure 2. PL/M-96 Software Package

230613-1

PL/M-96 COMPILER

FEATURES

Major features of the PL/M-96 compiler and programming language include:

Structured Programming

Programs written in PL/M-96 are developed as a collection of procedures, modules and blocks. Structured programs are easier to understand, maintain and debug. PL/M-96 programs can be made more reliable by clearly defining the scope of user variables (for example, local variables in a procedure). REENTRANT procedures are also supported by PL/M-96.

Language Compatibility

PL/M-96 object modules are compatible with all other object modules generated by Intel MCS-96 translators. Programmers may choose to link ASM-96 and PL/M-96 object modules together.

PL/M-96 object modules were designed to work with other Intel support tools for the MCS-96. The DEBUG compiler control provides these tools with symbolic information.

Data Types Supported

PL/M-96 supports seven data types for programmer flexibility in various logical, arithmetic and addressing functions. The seven data types include:

- BYTE: 8-bit unsigned number
- WORD: 16-bit unsigned number
- DWORD: 32-bit unsigned number
- SHORTINT: 8-bit signed number
- INTEGER: 16-bit signed number
- LONGINT: 32-bit signed number
- REAL: 32-bit floating point number

Another powerful feature are BASED variables. BASED variables allow the user to map more than one variable to the same memory location. This is especially useful for passing parameters, relative and absolute addressing, and memory allocation.

Data Structures Supported

Two data structuring facilities are supported by PL/M-96. The user can organize data into logical groups. This adds flexibility in referencing data.

- Array: Indexed list of same type data elements
- Structure: Named collection of same or different type data elements
- Combinations of Both: Arrays of structures or structures of arrays

Interrupt Handling

Interrupts are supported in PL/M-96 by defining a procedure with the INTERRUPT attribute. The compiler will generate code to save and restore the program status word when handling hardware interrupts of the MCS-96.

Compiler Controls

Compile time options increase the flexibility of the PL/M-96 compiler. These controls include:

- Optimization
- Conditional compilation
- The inclusion of common PL/M-96 source files from disk
- Cross reference of symbols
- Optional assembly language code in the listing file

Code Optimizations

The PL/M-96 compiler has four levels of optimization for reducing program size.

- Combination of constant expressions; "Strength reductions" (e.g.: a shift left rather than multiply by two)
- Machine code optimizations; elimination of superfluous branches; reuse of duplicate code, removal of unreachable code
- Overlaying of on chip RAM variables
- Optimization of based variable operations
- Use of short jumps where possible

Built in Functions

An extensive list of built in functions has been supplied as part of the PL/M-96 language. Besides TYPE CONVERSION functions, there are built in functions for STRING manipulations. Functions are provided for interrogating the MCS-96 hardware flags such as CARRY and OVERFLOW.

Error Checking

If the PL/M-96 compiler detects a programming or compilation error, a fully detailed error message is provided by the compiler. If a syntax or program error is detected, the compiler will skip the code generation and optimization passes. This powerful PL/M-96 feature can yield a two times increase in throughput when a user is in the initial program development cycle.

BENEFITS

PLM-96 is designed to be an efficient, cost-effective solution to the special requirements of MCS-96 Microcontroller Software Development, as illustrated by the following benefits of PL/M use:

Low Learning Effort

PL/M-96 is easy to learn and to use, even for the novice programmer.

Earlier Project Completion

Critical projects are completed much earlier than otherwise possible because PL/M-96, a structured high-level language, increases programmer productivity.

Lower Development Cost

Increases in programmer productivity translate immediately into lower software development costs because less programming resources are required for a given programmed function.

Increased Reliability

PL/M-96 is designed to aid in the development of reliable software (PL/M programs are simple statements of the program algorithm). This substantially reduces the risk of costly correction of errors in systems that have already reached full production status. The more simply the program is stated, the more likely it is to perform its intended function.

Easier Enhancements and Maintainance

Programs written in PL/M tend to be self-documenting, thus easier to read and understand. This means it is easier to enhance and maintain PL/M programs as the system capabilities expand and future products are developed.

ORDERING INFORMATION

Order Code	Operating Environment
D86PLM96	PL/M-96 Compiler for PC DOS 3.0 based Systems

Documentation Package

PL/M-96 User's Guide
MCS-96 Utilities User's Guide
MCS-96 Assembler and Utilities Pocket Reference Card
8096 Floating Point Arithmetic Library

SUPPORT

Hotline Telephone Support, Software Performance Report (SPR), Software Updates, Technical Reports, and Monthly Technical Newsletters are available.

C 96 SOFTWARE PACKAGE

- Implements the Full Programming Capabilities of the C Language
- Complies with Draft ANSI Standard
- Produces Relocatable Object Code which is Linkable to Object Modules Generated by Other MCS®-96 Translators
- Produces High-Density Code That Rivals Assembly in Efficiency
- Fully Linkable with the PL/M-96 and ASM-96 Programming Languages

- IEEE Floating Point Library (FPAL96) Included for Numeric Support
- Supports All of the Standard C Language I/O Library (STDIO)
- Includes a Linking and Relocating Utility, an Object-To-Hexadecimal Converter, and a Library Manager
- Supports the 80C196 Architecture

Intel's C 96 is a general purpose, structured programming language designed to support applications for the 16-bit family of MCS-96 microcontrollers including the 80C196. C 96 implements the C language as described in the Kernighan and Ritchie book, *The C Programming Language* (Prentice-Hall) Software Series, 1978). The latest enhancements to the C programming language as defined by the draft proposed ANSI C standard (e.g., structure assignments, and the void and enum data types) are supported.

The C 96 compiler translates C 96 language statements into MCS-96 machine instructions. The compiler generates code in Intel's relocatable Object Module Format (OMF) without using an intermediate assembly file. The OMF files can then be debugged using either the iSBE-96 emulator, the VLSiCE-96 emulator, or the ICE-196.

C 96 is available for the IBM PC AT and the PC XT

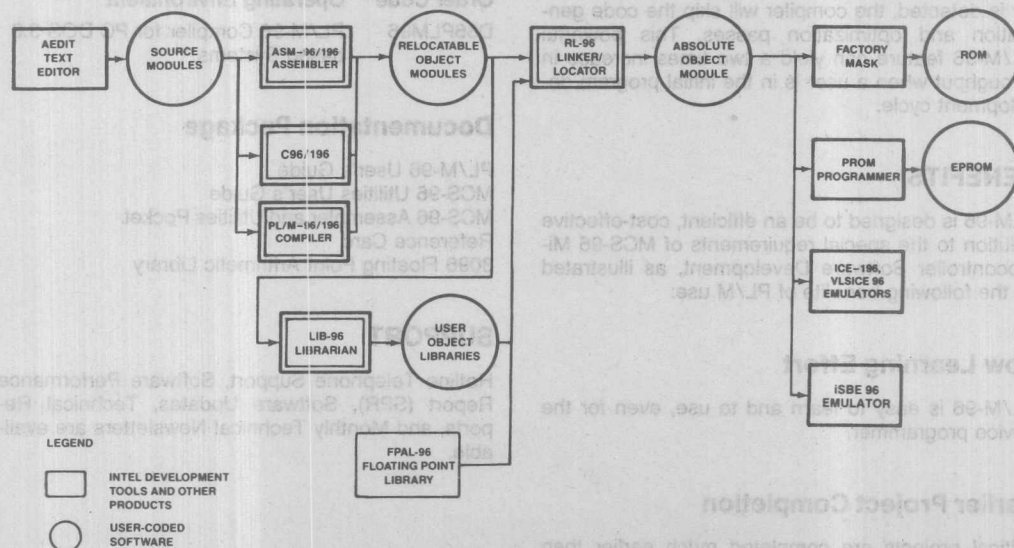


Figure 3. 8096 Software Development Process

C 96 COMPILER

COMPILER DESCRIPTION

Major features of the C 96 compiler include the preprocessor, the parser, and the code generator and optimizer. The code is output in Intel relocatable Object Module Format (OMF). The compiled code can then be debugged with either the iSBE-96 emulator or the VLSiCE-96 emulator.

The preprocessor interprets statements in the source code and performs such actions as macro expansion, file inclusion, and conditional compilation (for example, the #if directive, which specifies optional inclusion or exclusion of code).

The parser performs syntactic and semantic error checking on the code. The code generator converts the parser's output into efficient binary code. The optimizer streamlines the code and generates Intel relocatable OMF code, without creating an intermediate assembly file.

The compiler's DEBUG/NODEBUG control option specifies whether or not the object module should contain debug information. The debug information can be used to debug the compiled program using either the iSBE-96 emulator or the VLSiCE-96 emulator.

COMPILER FEATURES

Some of the features of the C 96 compiler are:

- Declarations
- Expressions and operators
- Statements
- Run-time library (STDIO)
- Compiler invocation
- Output conventions

Each of these features is discussed in the following sections.

DECLARATIONS

Declarations are used to specify the attributes of a set of identifiers. The scope of a declaration can encompass the entire source file or be local to a function body or block.

The storage class specifier defines the location and scope. The storage classes are as follows:

- auto active block
- extern external data definition
- static active data segment or register segment
- typedef a type definition (not storage allocation) that defines another name or a synonym

The storage class can be further defined with one of the following storage class modifiers:

- const code segment
- register machine register
- volatile I/O port (modifies the extern storage class only)

Identifiers are defined by their type. The types fall into one of the following categories:

- basic characters, integers, floating point numbers
- derived arrays, structures, unions, enumerations, functions, and pointers
- void empty set

The type is further defined by the following type specifiers:

- char, short, int, long, signed, unsigned, float, double, struct, union, enum, and typedef

EXPRESSIONS AND OPERATORS

All of the C language expressions and operators are supported by Intel's C 96 compiler. Table 1 is a summary of the C operators, arranged in order of precedence (from top to bottom). Operator precedence within an expression is evaluated in the order of associativity shown in Table 1.

STATEMENTS

A statement is a program element that specifies an action to be performed. The C language supports the following types of statements:

- Simple any valid expression
- Compound an optional list of variable declarations followed by a list of statements

- Selection an if or switch statement which is optionally included dependent on specified conditions
- Iteration a do, while or for statement which executes repeatedly until the controlling value is zero
- Branching a break, continue, goto, or return statement which changes the program control flow

Table 1. Precedence and Associativity

Class	Operator	Associativity
primary	[] () . →	left to right
unary	++ -- * + - ~ sizeof far	right to left
binary mult.	* / %	left to right
binary add	+ -	left to right
binary shift	<< >>	left to right
binary relat.	< > <= >=	left to right
binary equal.	= =	left to right
bitwise AND	&	left to right
bitwise XOR	^	left to right
bitwise OR		left to right
logical AND	&&	left to right
logical OR		left to right
conditional	? :	right to left
assignment	= *= /= %= += -= <<= + >>= &= ^= =	right to left left to right
comma	,	left to right

RUN-TIME LIBRARY (STDIO)

Intel's C 96 compiler supports the standard C language I/O library functions (STDIO). The include files listed in Table 2 are included with the C 96 compiler.

Table 2. C 96 Include Files

Name	Description
ctype.h	Used to declare and map characters.
errno.h	Used for error checking.
setjump.h	Used to bypass a normal call/return.
stdio.h	Used for standard I/O functions.
string.h	Used to manipulate strings.
time.h	Used to manipulate the time and date.

Character and arithmetic conversion functions are also included (atof, atoi, atol, csr, tolower, toupper, and uastr).

COMPILER INVOCATION

Intel's C 96 compiler is invoked with the following general syntax:

c96 pathname [controls]

The following invocation controls are some of the options supported by the C 96 compiler.

- Object file controls—DEBUG/NODEBUG, OBJECT, OPTIMIZE (0 through 3), REGISTERS, REGOVERLAY/NOREGOVERLAY, TYPE/NOTYPE
- Listing controls (selection and content)—CODE/NOCODE, COND/NOCOND, LIST/NOLIST, LISTINCLUDE/NOLISTINCLUDE, PREPRINT/NOPREPRINT, SYMBOLS/NOSYMBOLS, XREF/NOXREF
- Listing format controls—PAGING/NOPAGING, PAGELENGTH, PAGEWIDTH
- Source inclusion control—INCLUDE

The REENTRANT/NOREENTRANT extension has been added to the C 96 compiler invocation controls to enhance the compiler's use of the MCS-96 architecture. This extension enables the compiler to fully use the large register set of the MCS-96 family of microprocessors. When porting to programs in other environments, these keywords should be either removed or defined as null.

Output Conventions

The C 96 compiler produces a listing file and an object file. The listing file contains a formatted list of the source code and a list of compiler error messages. The compiler produces the object file in Intel's relocatable OMF code directly, without creating an intermediate assembly file.

BENEFITS

There are many benefits to the C 96 compiler, as explained in the following sections.

PROGRAM DEBUGGING

With the DEBUG control the C 96 compiler produces extensive debug information, including symbols. The debug information can be used to debug the program code with either the VLSICE-96 emulator or the iSBE-96 emulator. This serves to enhance programmer productivity.

FASTER COMPILATION

The C 96 compiler creates Intel object module format (OMF) directly, without creating an intermediate assembly file. This increases the compiler's execution speed.

PORTABLE CODE

Code portability has been designed into the C 96 compiler. The C 96 code is fully linkable with both the PL/M-96 and the ASM-96 programming languages.

Because the compiler supports the standard C library and produces Intel OMF code, programs developed on a variety of machines can be transported to the MCS-96. In addition, because C 96 conforms to accepted C language standards, programmers can quickly begin programming the MCS-96.

FULL MANIPULATION OF THE 8096 MICROCONTROLLER

The C 96 compiler has been highly optimized for the MCS-96 architecture. The REENTRANT/NOREENTRANT control has been added so that the compiler can identify non-reentrant procedures. This is extremely useful because it enables the programmer to have full access to the large MCS-96 register set.

With the C 96 compiler, the programmer can declare register variables that are not local to any procedure. Due to the large register set of the MCS-96 architecture, the compiler can dedicate registers to such variables.

SOFTWARE SUPPORT

Intel's Software Support Service provides maintenance on software packages with software support contracts which include subscription services, information phone support, and updates. Consulting services can be arranged for on-site assistance at the customer's location for both short-term and long-term needs. For more information, contact your local Intel Sales Office.

ORDERING INFORMATION

Part Number	Description
D86C96	C 96 Software Package
	PL/M-96 packages also include the RL96 Linker and Relocator, the FPAL96 Floating Point Library, and the LIB96 librarian utility.

Operating Environment

IBM PC AT
IBM PC XT

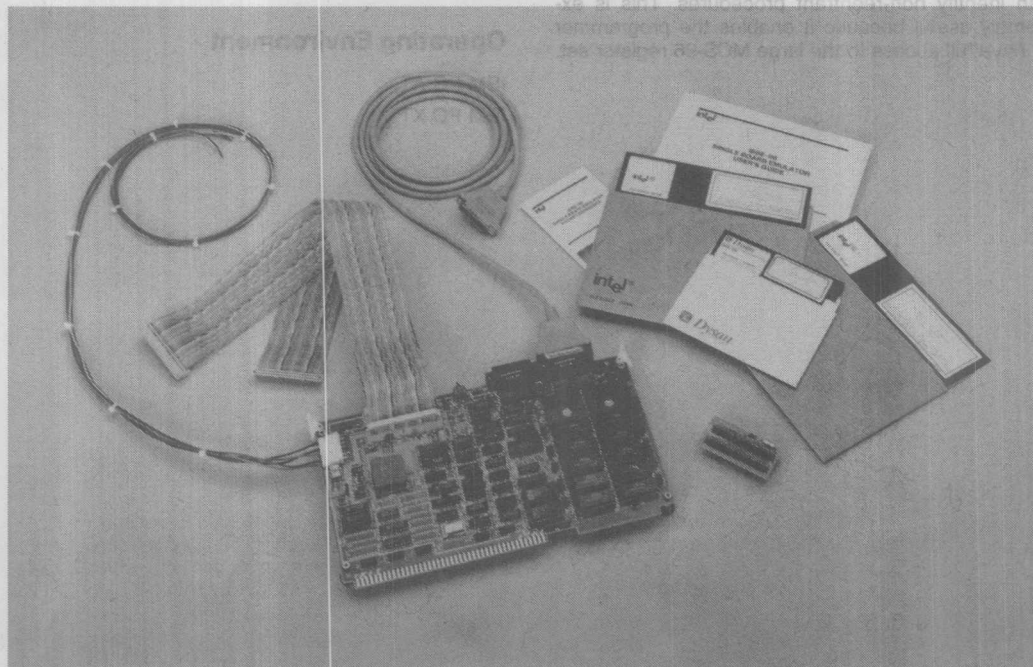


ISBE-96 DEVELOPMENT KIT SINGLE BOARD EMULATOR AND ASSEMBLER FOR THE MCS®-96 FAMILY OF MICROCONTROLLERS

- **Hosts**
 - Intel® Series III/IV Development Systems
 - IBM® PC AT, PC XT, and Compatibles (3.0)
- **Eight Software Execution Breakpoints That Can Selectively Be Turned On and Off**
- **12 MHz Emulation Speed**
- **Single Line Assembler/Disassembler**
- **MCS®-96 Software Support Package**
- **Configurable Serial I/O**
- **17.75 of On-Board User Memory**
- **Optionally Expandable to 64K of On-Board User Memory**

The iSBE-96 emulator supports the execution and debugging of programs for the MCS-96 family of microcontrollers at speeds up to 12 MHz. The MCS-96 family configurations are shown in Table 1. The iSBE-96 emulator consists of an 8097 microcontroller, a serial port and cables, and an EPROM-based monitor that controls emulator operation and the user interface.

The iSBE-96 emulator is a combination of hardware and software that permits programs written for the MCS-96 family of microcontrollers to be run and debugged in the emulator's artificial environment or in the user's prototype system. As a result, development time can be reduced by the early integration of hardware and software.



231015-1

FUNCTIONAL DESCRIPTION

Integrated Hardware and Software Development

The iSBE-96 emulator allows hardware and software development to proceed simultaneously. This approach is more time- and cost-effective than the alternate method: independent hardware and software development followed by system integration. With the iSBE-96 emulator, prototype hardware can be added to the system as it is designed; software and hardware integration occurs while the product is being developed. The emulator aids in the recognition of hardware and software problems.

Emulation is the controlled execution of the prototype software in the prototype hardware or in an artificial hardware environment that duplicates the microcontroller of the prototype system. The iSBE-96 emulator permits reading and writing of system memory, and control of program execution. The emulator also allows interactive debugging of the prototype software and can externally control program execution while operating in the prototype system. When the prototype system memory is not yet available, the iSBE-96 emulator's on-board memory permits software debugging.

Table 1. The Configurations of the MCS®-96 Family of Microcontrollers

		68 Pin	48 Pin
Digital I/O	ROMLESS	8096	
	ROM	8396	
	EPROM	8796	
Analog and Digital I/O	ROMLESS	8097	8095
	ROM	8397	8395
	EPROM	8797	8795

iSBE-96 Software

The iSBE-96 emulator software is available for use with the following host systems:

- Intellec Series III and Series IV development systems
- IBM PC/AT and PC/XT computer systems

The iSBE-96 emulator software is also available from U S Software* for use on the Intel Personal Development System (iPDS™) and the Intellec Series II development system.

***NOTE:**

U S Software is a registered trademark of United States Software Corporation.

The iSBE-96 emulator is supplied with a driver routine that communicates with the monitor software on the iSBE-96 emulator board through serial channel 1 or 2 (com1/com2). The driver interrupts the 8097 using the non-maskable interrupt (NMI) line for incoming keyboard input. The commands associated with the driver and the monitor are described in the following sections.

iSBE-96 Driver

iSBE-96 emulator is shipped with driver software for use on the Series III/IV development systems and the IBM PC AT/XT running PC DOS, version 3.0 or greater. The driver software provides a few easy-to-use commands. These commands are described in Table 2. ASM/DASM available on DOS version only.

Table 2. iSBE-96 Driver Commands

Driver Command	Function
ASM	Loads memory with MCS-96 assembly mnemonics.
DASM	Displays memory as MCS-96 assembly mnemonics.
EXIT	Exits the driver and returns to the host operating system.
<CONTROL> C	Same as for the EXIT command.
HELP	Displays the syntax of all commands.
INCLUDE	Specifies a command file.
<CONTROL> I	Turns the command file on and off.
<TAB>	Same as <CONTROL> I (turns the command file on and off).
LIST	Specifies a list file.
<CONTROL> L	Turns list file on and off.
<CONTROL> S	Stops scrolling of the screen display.
<CONTROL> Q	Resumes scrolling of the screen display.
<CONTROL> X	Deletes the line being entered.
<ESCAPE>	Aborts the command executing.

iSBE-96 MONITOR

The iSBE-96 monitor performs the following functions:

- Loads and saves user programs.
- Independently emulates user programs.

Monitor Command	Function
BAUD	Sets up the baud rate.
BR	Permits display and setting of up to eight software breakpoints.
BYTE	Permits display and changing of a single byte or range of bytes of memory or a single byte of the 8097 internal registers.
CHANGE	Permits display and changing of a series of memory words or bytes.
<CONTROL> S	Stops scrolling of the screen display.
<CONTROL> Q	Resumes scrolling of the screen display.
<CONTROL> X	Deletes the line being entered.
<ESCAPE>	Aborts the command executing.
GO	Begins emulation and continues until an enabled breakpoint is reached or the escape key is pressed.
LOAD	Loads programs and data from disks.
MAP	Permits mapping of several preprogrammed memory maps; also permits configurable serial I/O and selective servicing of the watchdog timer.
PC	Displays and changes the program counter.
PSW	Displays and changes the program status word.
RESET CHIP	Resets the 8096 to power-up conditions.
SAVE	Saves programs and data to disks.
SP	Displays and changes the stack pointer.
STEP	Provides single-step emulation with selective display formats.
VERSION	Displays the monitor version number.
WORD	Permits display and changing of a single word or range of words of memory or a single word of the 8097 internal registers.

- Examines registers.
- Maps the file capabilities of the serial ports (DS/DT).
- Maps different memory configurations.

The monitor commands are described in Table 3.

Integrating Hardware and Software

When the prototype system hardware is developed, the iSBE-96 emulator interfaces to the prototype through two 50-pin ribbon cables. The emulator can then execute code from the iSBE-96 on-board RAM (or from user-provided memory) and exercise the prototype system hardware.

BLOCK DIAGRAM

Figure 1 is a block diagram showing the iSBE-96 emulator. The following sections describe each block.

The Processor

The 68-pin processor of the iSBE-96 emulator is used only in the 8097 external-access mode. An 8097BH will be supported in 16-bit bus mode only.

An adapter board is provided for the 68-pin PGA version of the 8096 and 8097 microcontrollers. When debugging a 68-pin package, the two 50-pin ribbon cables plug into the 68-pin adaptor board which is plugged into the 68-pin socket on the prototype system.

When debugging a 48-pin package, the two 50-pin cables plug into the 48-pin adaptor board, which is then plugged into a 48-pin socket in the prototype system. A 68-pin PLCC Adaptor may be ordered.

iSBE-96 Emulator I/O

The iSBE-96 emulator's memory-mapped I/O devices are used to manage the system. These I/O devices are mapped into memory between locations 01F00H and 01FFFFH.

Included as part of the I/O are two serial ports. One is configured as data set (DS) and the other as data terminal (DT). When operating with an Intellec® development system, the data set port is used as the system console and the link for exchanging files.

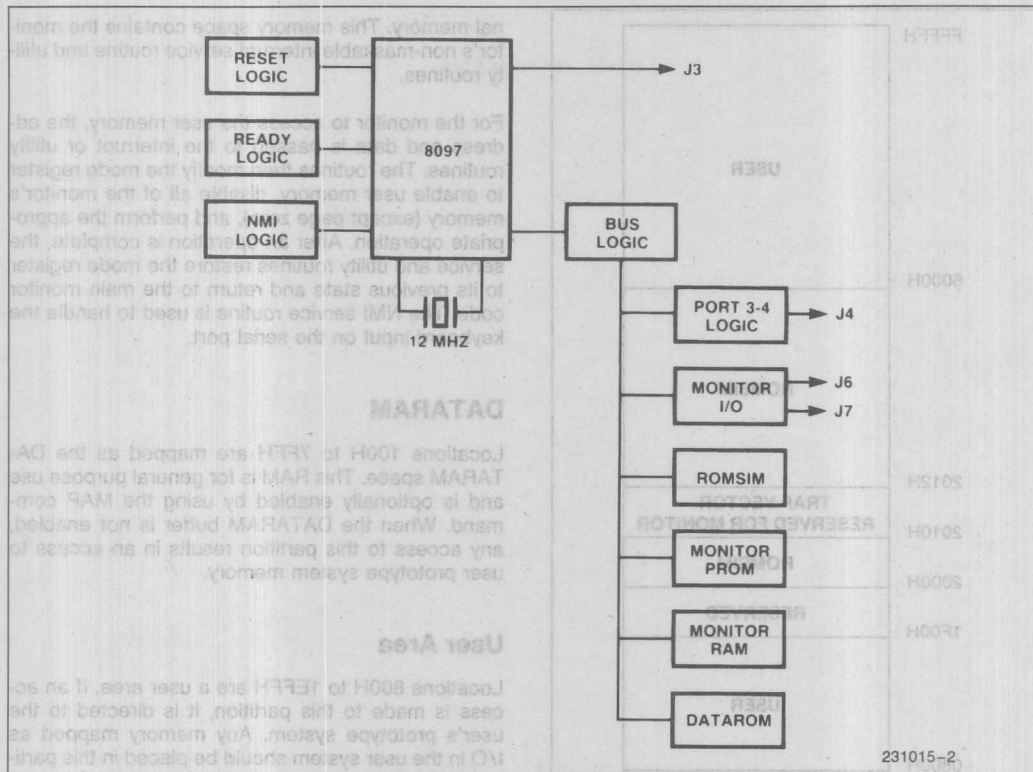


Figure 1. Block Diagram for the ISBE-96 Single Board Emulator

The serial ports are serviced under control of the NMI interrupt. The NMI interrupt has highest priority on the microcontroller and interrupts the user program when characters are entered from the keyboard. When in emulation, the monitor will still service inputs from the keyboard and execute certain monitor commands. Monitor activity is not transparent to the user.

Simulated ROM (ROMSIM)

There are eight 28-pin JEDEC byte-wide sockets with 2K-by-8 static RAMs present on the board. The partition on the user's prototype system that will be ROM is simulated by RAM on the ISBE-96 emulator board. This RAM facilitates easy program development, allowing users to correct and test problems in their programs.

ROMSIM can be expanded by replacing the ISBE-96 RAMs with 8K-by-8 static RAMs.

Port 3-4 Logic

The port 3-4 logic has two functions: to provide bus expansion and to provide I/O ports. The port 3-4 logic is controlled by a software switch available with the MAP command.

The ISBE-96 emulator reconstructs ports 3 and 4 of the 8395, 8396, and 8397 microcontrollers when the logic is defined by the MAP command as port 3-4. This port function should be selected when one of these four microcontrollers is intended as the target microcontroller.

When the BUS switch of the MAP command is specified, the ISBE-96 address/data expansion bus is available to the prototype system.

THE ISBE-96 EMULATOR MEMORY MAP

The target system should be designed with a memory map that is compatible with one of the ISBE-96

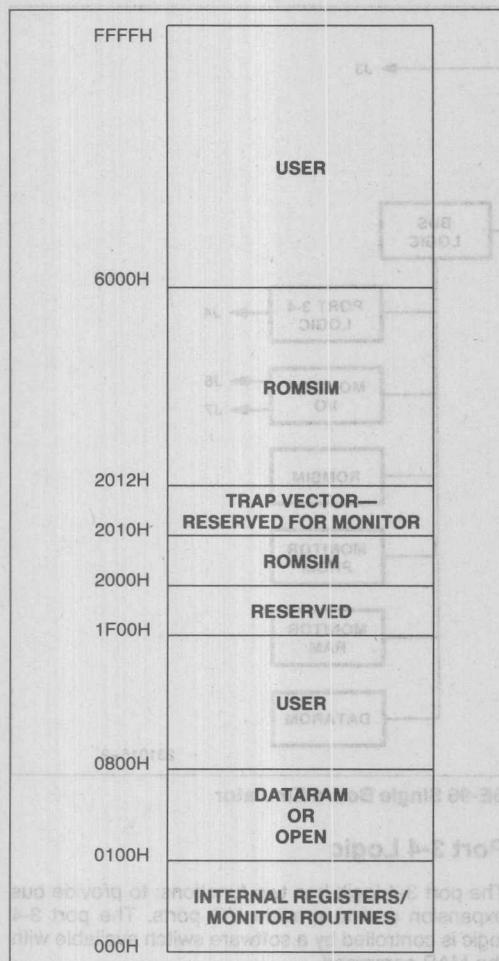


Figure 2. iSBE-96 Emulator Default Mapping

memory maps. Figure 2 shows the default address mapping. The following sections describe the areas of memory.

Internal Registers/Monitor Routines

Normally locations 000H through 0FFH contain the internal register space of the 8097. However, instruction fetches from these locations access exter-

nal memory. This memory space contains the monitor's non-maskable interrupt service routine and utility routines.

For the monitor to access the user memory, the address and data is passed to the interrupt or utility routines. The routines then modify the mode register to enable user memory, disable all of the monitor's memory (except page zero), and perform the appropriate operation. After an operation is complete, the service and utility routines restore the mode register to its previous state and return to the main monitor code. The NMI service routine is used to handle the keyboard input on the serial port.

DATARAM

Locations 100H to 7FFH are mapped as the DATARAM space. This RAM is for general purpose use and is optionally enabled by using the MAP command. When the DATARAM buffer is not enabled, any access to this partition results in an access to user prototype system memory.

User Area

Locations 800H to 1EFFH are a user area. If an access is made to this partition, it is directed to the user's prototype system. Any memory mapped as I/O in the user system should be placed in this partition. With 8K-by-8 static RAMs, this area is located and available on the iSBE-96 board.

Reserved Area

Locations 1F00H to 1FFFH are reserved by the monitor for on-board I/O devices.

ROMSIM

Because some of the MCS-96 family of microcontrollers are ROMLESS parts, a user program can be loaded for execution into the on-board RAMS of the iSBE-96 emulator. Locations 2000H to 5FFFH are mapped to this RAM space; the space is called ROMSIM.

Trap Vector

Locations 2000H to 2010H are the interrupt vector locations. Vector address location 2010H is used by the iSBE-96 monitor for breakpoints.

User Area

The partition 6000H to 0FFFFH is mapped to the user prototype area. During emulation any access to this partition is directed to the user's prototype system.

EXPANDING ON-BOARD MEMORY

On-board memory can be expanded to a full 64K bytes by replacing the supplied 2K-by-8 static RAMs with 8K-by-8 static RAMs or PROMs. The user may also replace on-board ROMSIM memory with 2K-by-8 PROMs or even locate all 64K bytes of memory on the prototype system.

DESIGN CONSIDERATIONS

Designers should note the following considerations for designing with the iSBE-96 emulator:

- The iSBE-96 software uses 6 bytes of user stack space.
- Analog signal accuracy is impaired when driven over the emulator cable (up to ± 50 mV loss of A/D conversion accuracy).

- The iSBE-96 emulator has some ac/dc parametric differences from the 8097 chip.
- The NMI vector is used for console service (Intel reserved interrupt).
- Keyboard activity during emulation affects real-time emulation because a 50 to 100 microsecond interrupt service routine is executed for every keystroke.
- The only hardware reset available for the iSBE-96 emulator is the system reset momentary switch (switch 1 on the emulator board).
- User system memory should be configured to the iSBE-96 memory map (see Figure 2).
- The iSBE-96 emulator does not support a user system crystal as shipped.
- The iSBE-96 driver software provided by Intel is not compatible with the Inteltec Model 800 or Series II Development Systems.
- The IBM PC/AT and PC/XT have been evaluated and accepted by Intel as compatible hosts for its development systems. Intel has not evaluated any other PC DOS machines (3.0). However, Intel knows of no reason why these PC DOS machines would not be compatible hosts for its development systems.

SPECIFICATIONS

Equipment Supplied

Standard MULTIBUS®-size board assembly

EPROM-based monitor

Auxiliary power cable

RS-232 serial cables

Two standard, 18 in., 50-pin ribbon cables for connection to the user's prototype system

Adapter board for the 48-pin DIP and 68-pin PGA versions of the MCS-96 microcontroller

MCS-96 software support package

One 8 in. single-density software disk for the Series III

One 8 in. double-density software disk for the Series III

One 5¼ in. software disk for the Series IV

One 5¼ in. software disk for the IBM PC AT/XT

Documentation

iSBE-96 User's Guide (Order number 164116)

iSBE-96 Pocket Reference (Order number 164157)

Developing MCS-96 Applications Using iSBE-96 (Order Number 280249-001, AP-273)

Emulation Clock

12 MHz supplied crystal

Physical Characteristics

Width: 6.75 in. (17.15 cm)

Length: 12 in. (30.48 cm)

Height: 0.75 in. (1.91 cm)

DC Electrical Requirements

Voltage	Current
+5V \pm 5%	3.5a max
+12V \pm 5%	0.06a max
-12V \pm 5%	0.05a max

Environmental Characteristics

Operating Temperature: 10°C to 40°C

Operating Humidity: 10% to 85% relative humidity, without condensation

IBM PC XT/AT Host Requirements

- PC DOS, version 3.0 or greater
- External power supply
- Serial channel Com1/Com2

ORDERING INFORMATION

Intel 3065 Bowers Ave.
Santa Clara, CA 95051

Part Number Description

SBE96SKIT iSBE-96 single board emulator for use with the Series III/IV development systems. The kit contains the following parts:

- iSBE-96 single board emulator
- MCS-96 software support package for the Series III/IV development systems
- iSBE-96 Series III/IV upgrade kit (cables and software needed to run on Intel Hosts)

SBE96DKIT iSBE-96 single board emulator for use with the IBM PC/AT and PC/XT computer systems. The kit contains the following parts:

- iSBE-96 single board emulator
- MCS-96 software support package for PC DOS
- iSBE-96 DOS upgrade kit (cables and software needed to run on the IBM PC/AT or PC/XT)

SBE96DU

iSBE-96 DOS upgrade kit for those customers who wish to upgrade their Series III/IV kit to run on the IBM PC AT or PC XT.

SBE96SU

iSBE-96 Series III/IV upgrade kit for those customers who wish to upgrade their DOS kit to run on Intel Hosts).

TASBEE

68-pin PLCC Adaptor Board.

U S Software

5470 N. W. Innisbrook
Portland, OR 97229

Phone: 503-645-5043

International Telex 4993875

Part Number Description**XASM96**

Performs assembly of MCS®-96 programs written on the iPDS.

ATOP96

iPDS and Series II software for iSBE-96 host communications. Performs host communications and assembly/disassembly of iSBE-96 instructions. The XASM Host Cross Assembler software must be ordered with this software.



128E-96

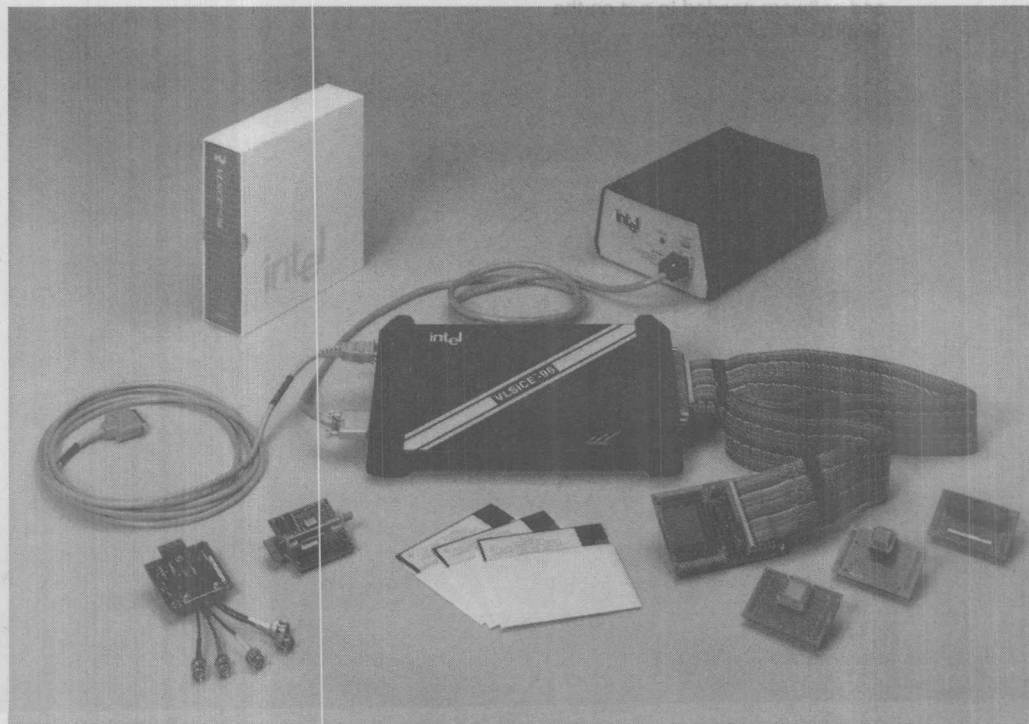


VLSICE-96

IN-CIRCUIT EMULATOR FOR THE 8X9X FAMILY OF MICROCONTROLLERS

- Precise Real-Time Emulation of the 8X9X Family of Components
- 64K of Mappable Memory for Early Software Debug and (EP)ROM Simulation
- A 4K-Entry Trace Buffer for Storing Real-Time Execution History, Including Both Code and Data Flows
- Fastbreaks and Dynamic Trace
- Symbolic Debugging Allows Accesses to Memory Locations and Program Variables (Including Dynamic Variables) Using Program-Defined Names from the User's Assembler or Compiler Source Code
- Shadow Registers Allow Reading Many 8096 Write-Only and Writing Many Read-Only Registers
- Break and Trace are Qualified on Execution Addresses, Data Addresses, and Values (Both External and Internal RAM), Opcodes, Selected PSW Flags, and 2 External Sync Lines
- Equipped with the Integrated Command Directory (ICD™) Which Provides
 - An On-Line Help File
 - A Dynamic Syntax Menu
 - Dynamic Command-Entry
 - Error Checking
 - On-Line Editor
- Serially Linked to Intel Series III/IV Hosts or IBM* PC-XT and AT

The VLSICE-96 In-Circuit Emulator is a debugging and test tool used for development of the hardware and software of a target system based on the 8X9X family of microcontrollers (8095, 8096, 8097, 8395, 8396, 8397, 8795, 8796, 8797, 8098, 8398, 8798) including BH components.



*IBM is a trademark of International Business Machines.

280140-1

INTRODUCTION

The VLSiCE-96 emulator allows hardware and software development of a design project to proceed simultaneously. With the VLSiCE-96 emulator, prototype hardware can be added to the system as it is designed and software can be developed prior to the completion of the hardware prototype. Thus, software and hardware can be integrated while the product is being developed.

The VLSiCE-96 emulator assists four stages of development:

- Software development
- Hardware development
- System integration
- System test

Software Development

The VLSiCE-96 emulator can be operated without being connected to a prototype or before any of the prototype hardware is available. In this stand-alone mode, the VLSiCE-96 emulator can be used to facilitate application program development.

Hardware Development

Because the VLSiCE-96 emulator precisely matches the component's electrical and timing characteristics as well as full bus access, it is a valuable tool for hardware development and debug.

System Integration

Integration of software and hardware begins when the microcontroller socket is connected to any functional element of the target system. As each section of the user's hardware is completed, it is added to the prototype. Thus, each section of the hardware and software can be system tested with the VLSiCE-96 emulator in real-time operation as it becomes available.

System Test

When the prototype is complete, it is tested with the final version of the system software. The VLSiCE-96 emulator can then be used to verify or debug the target system as a completed unit.

By providing support for the ROMLESS, ROM, and EPROM versions of the microcontroller, the VLSiCE-96 emulator has the ability to debug a prototype or production product at any stage in its development without introducing extraneous hardware or software test tools.

opment without introducing extraneous hardware or software test tools.

PHYSICAL DESCRIPTION

The VLSiCE-96 emulator consists of the following components (see Figure 1):

- Software (includes the VLSiCE-96 emulator software, diagnostic software, and tutorial)
- 68-pin PGA adaptor
- 68-pin PLCC adaptor (optional)
- 48-pin DIP adaptor (optional)
- Controller pod
- User cable assembly (consisting of the user cable and processor module)
- Serial cable (host-specific)
- Crystal power accessory (CPA)
- Multi-synchronous accessory (MSA) (optional)
- Power supply and V_{CC} booster module
- AC and DC power cables

VLSiCE-96 software fully supports all mnemonics, object file formats, and symbolic references generated by Intel's ASM-96, PL/M-96, and C-96.

The on-line tutorial is written in VLSiCE-96 command language. Thus, the user is able to interact with and use the VLSiCE-96 emulator while executing the tutorial.

The controller pod contains 64K of ICE memory, a 4K-entry trace buffer, and circuitry that provides communication between the host and the emulator.

The processor module contains a special version of the Intel 8096 microcontroller, called the emulation processor. This chip performs real-time and single-step execution of a program's object code for execution and debugging purposes in place of the target system microcontroller.

The crystal power accessory (CPA) is a small detachable board that connects to the back of the controller pod and is used to run the VLSiCE-96 emulator in the stand-alone mode. It is also used when running customer confidence tests. In the stand-alone mode, the user plug on the user cable is connected through the 68-pin PGA adaptor to the CPA. The CPA supplies clock and power. Stand-alone mode is used to test and debug software prior to the availability of hardware.

The optional multi-synchronous accessory can be used to connect the VLSiCE-96 emulator with up to 20 multi-ICE compatible emulators together for synchronous GO and BREAK emulation, and TRACE.

It can also be used with other debug equipment such as logic analyzers and oscilloscopes for synchronous GO, BREAK and TRACE.

The serial cable connects the host system to the controller pod. The serial cable has electrical specifications similar to the RS-232C standard.

The power supply connects to the controller pod via the V_{CC} booster module and the DC power cable. There are several voltage options available for the power supply depending on switch settings on the back of the power supply.

A comprehensive set of documentation is included with the VLSiCE-96 emulator.

Figure 1 shows a drawing of the VLSiCE-96 emulator.

VLSiCE-96 EMULATOR FEATURES

The VLSiCE-96 emulator assists hardware and software design engineers in developing, debugging and testing designs incorporating the 8X9X family of microcontrollers. The following are some of the VLSiCE-96 features:

Emulation

Emulation is the controlled execution of the prototype software in the prototype hardware or in an artificial hardware environment that duplicates the microcontroller of the target system. With the VLSiCE-96 emulator, emulation is a transparent process that happens in real-time without sacrificing microcontroller resources. The execution of prototype software is facilitated through the VLSiCE-96 command language.

Memory Mapping

There are 64 Kbytes of zero-waitstate, high-speed mappable memory available. This memory space can be mapped to either the target system or to the on-board VLSiCE-96 memory space in 1 Kbyte blocks on 1 Kbyte boundaries. Mapping memory to the VLSiCE-96 emulator allows software development to proceed before prototype hardware is available. Memory mapping also gives the VLSiCE-96 emulator the capability to simulate the 8 Kbytes of (EP)ROM on those versions of the chip for code verification and validation.

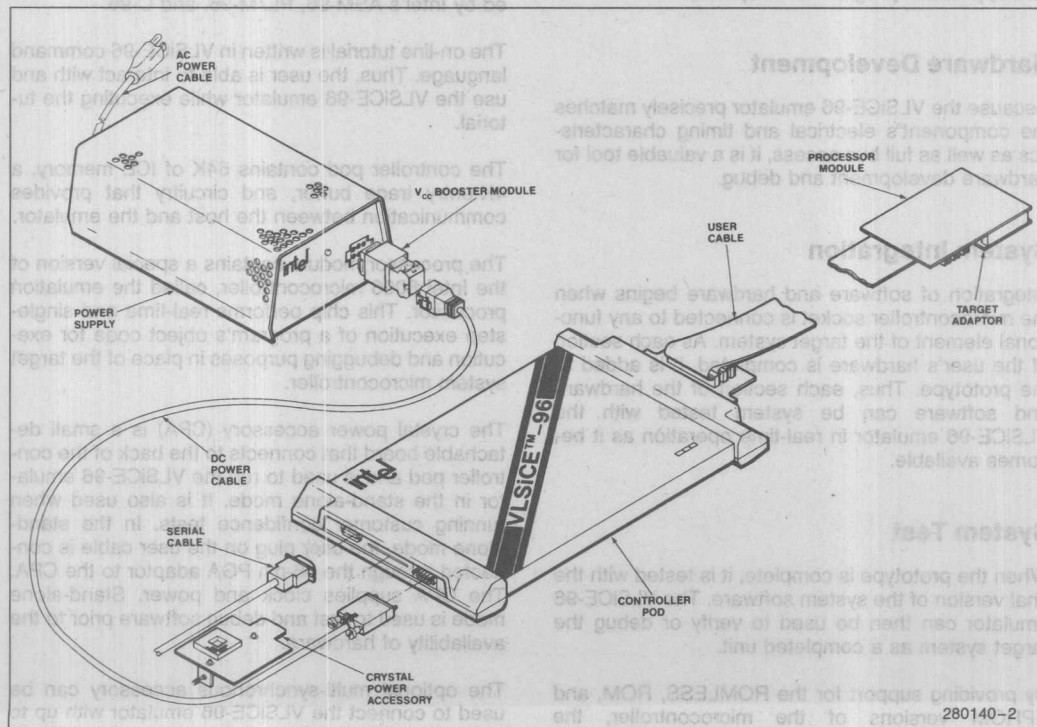


Figure 1. The VLSiCE™-96 Emulator

Memory Examination and Modification

The memory space for the 8X9X component and its target hardware is accessible through the emulator. The VLSiCE-96 software allows the component's special function registers to be accessed mnemonically (e.g. AD_RESULT, INT_MASK). A significant benefit to the VLSiCE-96 is its ability to read many of the write-only registers (e.g. ICC0, PWM_CONTROL) and to write many of the read-only registers (e.g. AD_RESULT, SBUFRX).

Data can be displayed or modified in several bases: hex, decimal, and binary, and in standard formats including: ASCII, real and integer. Program code can be disassembled and displayed as assembler mnemonics. It also can be modified with standard assembler statements.

Memory locations can also be examined or modified by their symbolic references. A symbolic reference is a procedure name, line number, or label in the user program that corresponds to a memory location.

Some typical symbolic functions include:

- Changing or inspecting the value of a program variable by using its symbolic name, rather than the address of the memory location.
- Defining break and trace events using symbolic references.
- Referencing static variables, dynamic (stack-resident) variables, based variables, and record structures combining primitive data types. The primitive data types are ADDRESS, BOOLEAN, BYTE, CHAR (character), WORD, DWORD (double word), INTEGER, LONGINT, SHORTINT, and REAL.

The VLSiCE emulator maintains a virtual symbol table for program symbols making it possible for the

table to exist without fitting entirely into host RAM memory. The size of the virtual symbol table is constrained only by the capacity of the disk.

Breakpoint Specifications

Breakpoints allow halting of a user program in order to examine the effect of the program's execution on the target system. Breakpoints can be defined as execution addresses, data addresses and data values (both external and internal RAM), opcodes, selected bits of the PSW flag, and as 2 external inputs (SYNCOIN AND SYNC1IN). These breaks can be arranged to occur over a range of addresses and with up to 8 levels of arming and disarming. After a break the user program can resume execution from where it left off.

Trace Specifications

Tracing can be triggered with the same conditions set for breaking. The trace buffer is displayed as disassembled instructions, data fetches and stores, and with the timetag showing the relative time at which the program executed each instruction. Figure 2 shows a trace display as a result of the PRINT command.

Normally, the VLSiCE-96 emulator traces program activity while the user program executes. With a trace specification, tracing can be specified to occur only when specific conditions are met during execution. The trace buffer collects data for up to 4 Kbyte entries of information during emulation.

The trace buffer can be examined during halt mode or if non-stop emulation is desired; the trace can be examined while emulation continues. If this second option is selected, trace collection stops while the trace buffer is uploaded to the host.

hlt>PRINT CYCLES NEWEST 8			
FRAME	ADDRESS	CODE	MNEMONIC OPERANDS TIME
(0017)	2086H	18EF80	SHRB 80H,EFH 5221 US
		[00EFH]= A3H(R)	[0080H]= 00H(R) [0080H]= 00H(W)
(0018)	2089H	00FF	SKIP FFH 5222 US
(0019)	208BH	FF	RST 5223 US
(0020)	2080H	E70000	LJMP \$+0003H 5233 US
(0021)	2083H	090000	SHL R0,#0H 5225 US
		[0000H]=0000H(R)	[0000H]=0000H(W)
(0022)	2086H	18EF80	SHRB 80H,EFH 5236 US
		[00EFH]= A3H(R)	[0080H]= 00H(R) [0080H]= 00H(W)
(0023)	2089H	00FF	SKIP FFH 5237 US
(0024)	208BH	FF	RST 5238 US

Figure 2. The Trace Buffer Display

Arming and Triggering

The VLSiCE-96 command language allows specification of complex events with up to 8 states, each with several conditions. For example, a specification can be made that causes a break to occur when a variable is written only within a certain procedure. The execution of the procedure is the arm condition and the variable modification is the break condition. The arm condition is an optional part of a break/trace sequence in the VLSiCE emulator. A set of arm conditions can be used to ensure that a break is not possible until all required qualifying conditions are satisfied.

Procedures

Debugging procedures (PROCS) are a user-named group of VLSiCE commands that are executed sequentially. Procs can simulate missing hardware or software, collect debug information, and make troubleshooting decisions. They can be copied to text files on disk, then included from the file into the command sequence in later test sessions.

Procedures can also serve as programmable diagnostics, implementing new emulator commands for special purposes.

FASTBREAKS

Fastbreaks make it possible to examine and modify memory without halting emulation. The commands that can be executed are simple one-access functions, such as, WORD 1FH or IOS0. When enabled, fastbreaks occur whenever a memory access is made.

Breakpoints and tracepoints can be re-specified during emulation with fastbreaks enabled.

While emulation does not halt during fastbreaks, a delay in code execution occurs when a fastbreak is requested. In most cases, this latency in code execution is less than 150 μ s.

Interrupts During Interrogation (IDI) Mode

The VLSiCE-96 software can service and record interrupts even though emulation has been halted (interrogation mode). In the special mode designated

as IDI mode, hardware interrupts can be serviced while the emulator is being interrogated. Use of this mode is determined by the setting of a VLSiCE-96 pseudo-variable (IDI_PC). After breaking from emulation or fastbreaks mode, whenever an interrupt occurs, the processor jumps to the appropriate vector and executes the interrupt service routine.

The setting of another VLSiCE-96 pseudo-variable (INT_REC_EN) allows the recording of interrupts but not the servicing of interrupts, during halt mode. If the pseudo-variable is set to TRUE, all interrupts are recorded in the INT_PENDING register, and serviced when the emulator re-enters emulation.

Dynamic Tracing

The trace buffer can be accessed in two ways, dynamically during emulation and statically after emulation halts. While dynamically tracing, any form of the PRINT command can be entered and the specified portion of the trace buffer is displayed. This allows real-time display of processor activity. Displaying the trace buffer during emulation stops collection of trace and some trace information can be lost, but emulation is unaffected.

On-Line Syntax Guide

A special syntax guide called the Integrated Command Directory (ICD), at the bottom of the display screen, aids in creating syntactically correct command lines. Figure 3 shows an example of the ICD for the GO command.

HELP

This feature provides assistance with the emulator commands through the host terminal. HELP is available for most of the commands. Figure 4 shows help for one of the commands.

Multi-Synchronous Operation

The VLSiCE-96 emulator can run with other emulators, and lab equipment such as logic analyzers or oscilloscopes. VLSiCE-96 emulators can be daisy-chained together in a network to work simultaneously to test a prototype system. The multi-synchronous operation is facilitated by the optional multi-synchronous accessory.

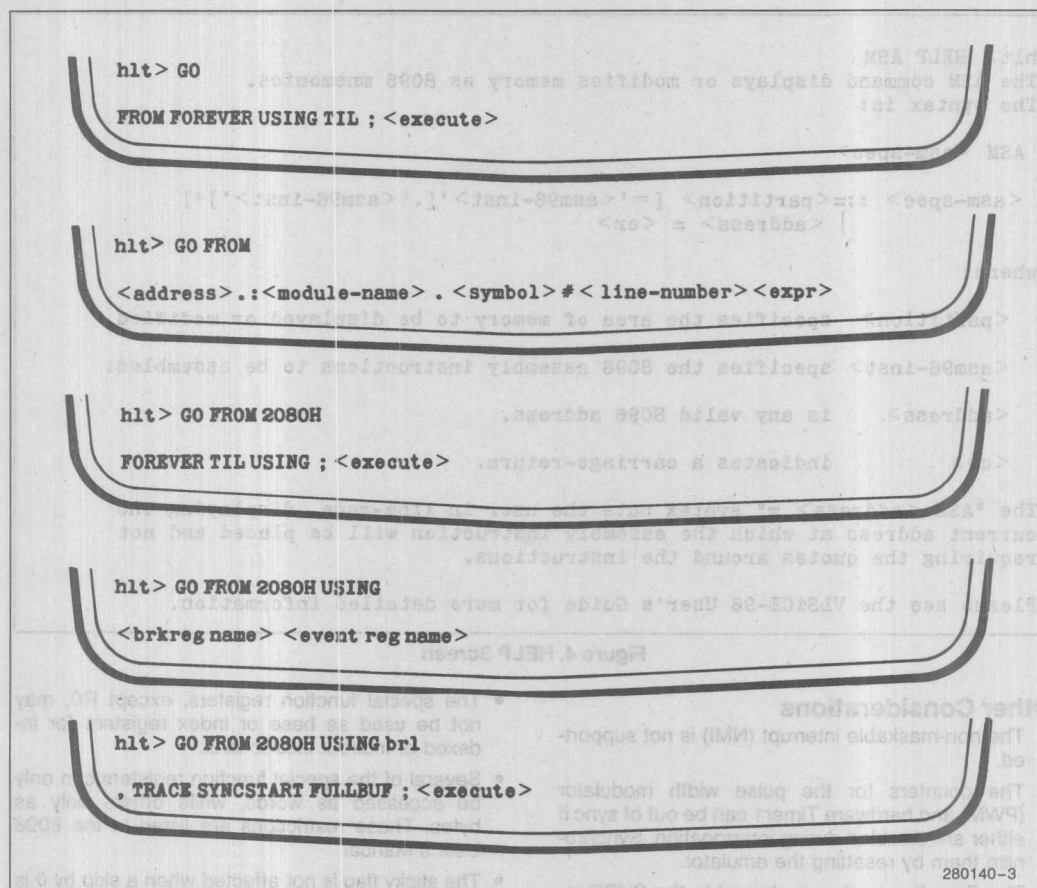


Figure 3. The Integrated Command Directory for the GO Command

DESIGN CONSIDERATIONS

There are design considerations to be aware of before designing with the VLSICE-96 emulator.

Electrical Considerations

The user pin timings, thresholds, and loadings are identical to the 8096 and 8098 components except the RESET and CLKOUT pins have an additional loading of 1 μ A and 10 pF. The Non-Maskable Interrupt (NMI) is not supported.

	Min.	Max.
Clock Frequency	6 MHz	10 MHz
V _{CC}	Emulator does not require system power to operate.	
I _{CC}		0 mA

Mechanical Considerations

The user plug is on the end of a three foot (1m) flexible cable. Adequate spacing must be provided on the target system to allow the emulation processor board and user plug to be inserted into the target system.

The height of the user plug should be considered for multiple board system prototypes that need to be debugged and tested. Be sure that the space between the boards is greater than 1½" (4 cm) to allow for the user plug.

Figure 5 shows the user plug dimensions. The user plug comprises the emulator processor board and the 68-pin or 48-pin adaptor. In the figure, please note the location of pin 1 on each adaptor.

```

hlt> HELP ASM
The ASM command displays or modifies memory as 8096 mnemonics.
The syntax is:

ASM <asm-spec>

<asm-spec> ::= <partition> [= '<asm96-inst>' ['<asm96-inst>']*]
              | <address> = <cr>

```

where:

- <partition> specifies the area of memory to be displayed or modified.
- <asm96-inst> specifies the 8096 assembly instructions to be assembled.
- <address> is any valid 8096 address.
- <cr> indicates a carriage-return.

The "ASM <address> =" syntax puts the user in line-mode, displaying the current address at which the assembly instruction will be placed and not requiring the quotes around the instructions.

Please see the VLSiCE-96 User's Guide for more detailed information.

Figure 4. HELP Screen

Other Considerations

- The non-maskable interrupt (NMI) is not supported.
- The counters for the pulse width modulator (PWM) and hardware Timer1 can be out of sync if either are disabled during interrogation. Synchronize them by resetting the emulator.
- The Zero flag is always cleared in the SUBC instruction. Therefore, the relational operators <= and > for LONG variables in C96 V1.0 and LONGINT variables in PL/M-96 V1.1, work incorrectly. These languages have been tailored for the 8X9X-90 microcontroller which either sets or resets the Zero flag in the SUBC instruction.

If there is a memory-resident program that is permanent on the PC, use of the DOS shell escape may corrupt the VLSiCE-986 software. To insure reliability, do not use the system escape on host systems that have permanent memory-resident programs.

The VLSiCE-96 emulator has some properties that are inherent in the 8X9XBH component. These are:

- Neither the source nor the destination address of the Multiply or Divide instructions can be a writable special function register.

- The special function registers, except R0, may not be used as base or index registers for indexed or indirect instructions.
- Several of the special function registers can only be accessed as words, while others only as bytes. These restrictions are listed in the 8096 User's Manual.
- The sticky flag is not affected when a skip by 0 is executed.
- To emulate the 8X9X-90 microcontroller, memory location 2018H in both target system emulator mapped memory should be 0FFH.
- The JBS and JBC instructions cannot be used directly on Port 2.1.

SPECIFICATIONS

Host Requirements

An IBM PC XT or PC AT with 512 Kbytes RAM and hard disk. Intel recommends and IBM PC AT with 640 Kbytes of RAM, one floppy drive and one hard disk, running PC-DOS 3.1 or later.

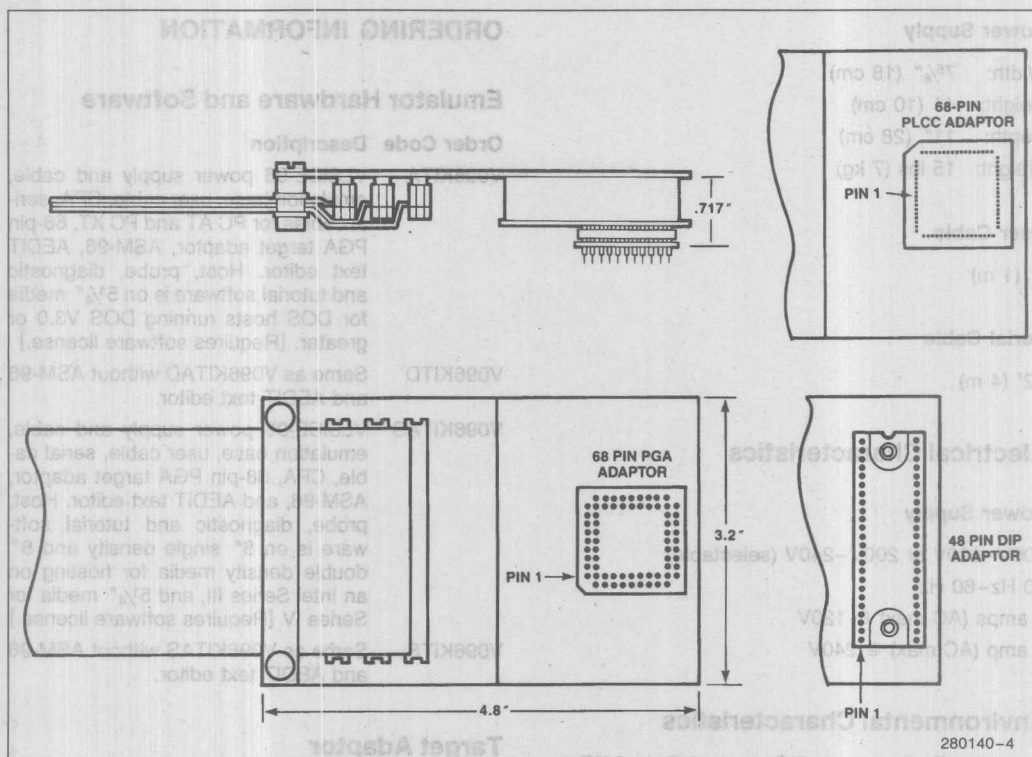


Figure 5. Dimensions for the Emulator Processor Board and Adaptors

An Intel® Microcomputer Development System, Series III or Series IV, running under ISIS or INDX, with at least 512 Kbytes of application memory resident, with dual floppy or one hard disk and one floppy drive required.

VLSICE-96 Software Package

VLSICE-96 emulator software

VLSICE-96 confidence tests

VLSICE-96 tutorial software

System Performance

Mappable zero wait-state (up to 10 MHz). Min 0 Kbytes Max 64 Kbytes Mappable to user memory or ICE memory in 1K blocks on 1K boundaries.

Trace Buffer 4 Kbytes × 48 bits

Virtual Symbol Table A maximum of 61 Kbytes of host memory space is available for the virtual symbol table (VST). The rest of the VST resides on disk and is paged in and out of host memory as needed.

Physical Characteristics

Controller Pod

Width: 8 1/4" (21 cm)

Height: 1 1/2" (4 cm)

Depth: 13 1/2" (34 cm)

Weight: 4 lbs (2 kg)

Power Supply

Width: 7 $\frac{5}{8}$ " (18 cm)

Height: 4" (10 cm)

Depth: 11" (28 cm)

Weight: 15 lbs (7 kg)

User Cable

3' (1 m)

Serial Cable

12' (4 m)

Electrical Characteristics

Power Supply

100V–120V or 200V–240V (selectable)

50 Hz–60 Hz

2 amps (AC max) @ 120V

1 amp (AC max) @ 240V

Environmental Characteristics

Operating Temperature: 0°C to +40°C (+32°F to +104°F)

Operating Humidity: Maximum of 85% relative humidity, non-condensing

DOCUMENTATION

VLSiCE-96 In-Circuit Emulator User's Guide, order number 165814

VLSiCE-96 In-Circuit Emulator Pocket Reference, order number 165815

VLSiCE-96 In-Circuit Emulator Installation Supplement, order number 166477

VLSiCE-96 Emulator Tutorial Guide, order number 165816

Debug Editor User's Guide, order 167098

ORDERING INFORMATION

Emulator Hardware and Software

Order Code Description

V096KITA VLSiCE-96 power supply and cable, emulation base, user cable, CPA, serial cables for PC AT and PC XT, 68-pin PGA target adaptor, ASM-96, AEDIT text editor. Host, probe, diagnostic and tutorial software is on 5 $\frac{1}{4}$ " media for DOS hosts running DOS V3.0 or greater. [Requires software license.]

V096KITD Same as V096KITAD without ASM-96 and AEDIT text editor.

V096KITAS VLSiCE-96 power supply and cable, emulation base, user cable, serial cable, CPA, 68-pin PGA target adaptor, ASM-96, and AEDIT text editor. Host, probe, diagnostic and tutorial software is on 8" single density and 8" double density media for hosting on an Intel Series III, and 5 $\frac{1}{4}$ " media for Series IV [Requires software license.]

V096KITS Same as V096KITAS without ASM-96 and AEDIT text editor.

Target Adaptor

Order Code Description

TA096E Optional 68-pin PLCC Adaptor board

TA096B Optional 48-pin DIP Target Adaptor board.

Multi-Synchronous Accessory

Order Code Description

MSA96 Optional Multi-Synchronous Accessory for multi-ICE capability.

Software Only
Order Code Description

SA096D	Software for host, probe, diagnostic and tutorial on 5 1/4" media for use with the PC AT and PC XT under PC-DOS V3.0 or greater. [Requires software license.]
SA096SD	Software for host, probe, diagnostic and tutorial on 5 1/4" media for use with the PC AT and PC XT under PC-DOS V3.0 or greater. [Requires software license.]
SA096S	Software for host, probe, diagnostic and tutorial on 8" single density and 8" double-density media for use with a Series III, and 5 1/4" media for use with a Series IV. [Requires software license.]

D86ASM96NL

ASM/R&L 96 for PC-DOS. It contains a macro assembler, a linker/locator utility, a floating point utility and a librarian. System requirements are an IBM PC AT or PC XT with 512 Kbytes of RAM and PC-DOS 3.0 or greater.

D86PLM96NL

PL/M 96 and R&L for PC-DOS. It contains a compiler, a linker/locator utility, a floating point utility and a librarian. System requirements are an IBM PC AT or PC XT with 512 Kbytes of RAM and PC-DOS 3.0 or greater.

D86C96NL

C96 and R&L for PC-DOS. Contains a compiler linker/locator utility, and all standard C libraries including STDIO. System requirements are an IBM PC AT or PC XT with 512 Kbytes of RAM and PC-DOS 3.0 or greater.

pSBE96SKIT

iSBE-96 single board emulator for use with the Series III/IV development systems. The kit contains:

iSBE-96 single board emulator

iSBE-96 Series III/IV upgrade kit (cables and software needed to run on Intel Hosts).

pSBE96DKIT

iSBE-96 single board emulator for use with the IBM PC AT and PC XT computer systems. The kit contains: iSBE-96 single board emulator 8096 software support package for PC-DOS, iSBE-96 DOS upgrade kit (cables and software needed to run on the IBM PC AT or PC XT).

Other Useful Intel 8X9X Debug and Development Support Products
Order Code Description

I86ASM96	Consists of the ASM 96 macro assembler that translates symbolic assembly language mnemonics into relocatable object code, and the RL96 linker and relocator program that links modules generated by ASM 96 and PL/M 96 and locates the linked object modules to absolute memory locations. System requirements and Intellec System running INDX.
I86PLM96	Consists of the PL/M 96 compiler that provides high level programming language support, the LIB 96 utility that creates and maintains libraries of software object modules, the FPAL96 floating point arithmetic library, and the RL96 linker and relocater program that links modules generated by ASM 96 and PL/M 96 and locates the linked object modules to absolute memory locations. System requirements and Intellec System running INDX.

Running the iSBE-96 emulator on the Series II and iPDS system requires software from:

U.S. Software Corporation
5470 N.W. Innisbrook
Portland, OR 97229
Phone: 503-645-5043
International Telex: 4993875

REAL-TIME TRANSPARENT 80C196 IN-CIRCUIT EMULATOR**REAL-TIME TRANSPARENT 80C196 IN-CIRCUIT EMULATOR**

The ICE-196PC in-circuit emulator delivers real-time high-level debugging capabilities for developing, integrating and testing 80C196-based designs. Operating at the full speed of the 80C196 microcontroller, the ICE-196PC provides precise I/O pin timings and functionality. The ICE-196PC also allows you to develop code before prototype hardware is available. The ICE-196PC in-circuit emulator represents a low-cost development environment for designing real-time microcontroller-based applications with minimal investment in time and resources.

ICE™-196PC IN CIRCUIT EMULATOR FEATURES

- Real-Time Emulation of the 80C196 Microcontroller
- 64K Bytes of Mappable Memory
- 2K-entry Trace Buffer
- 3 Breakpoints or 1 Range Break
- Symbolic Support and Source Code Display
- Standalone Operation
- Versatile and Powerful Host Software
- Hosted On IBM PC XT, AT* or Compatibles With DOS 3.0 or Later

REAL-TIME EMULATION

The ICE-196PC provides real-time emulation with the precise input/output pin timings and functions across the full operating frequencies of the 80C196 microcontroller. The ICE-196PC connects to the intended 80C196 microcontroller socket via a 16" flex cable, which terminates in a 68-pin PLCC probe.

MAPPABLE MEMORY

The ICE-196PC has 64K bytes (65,536) of zero wait-state memory that can be enabled or mapped as read-only, write-only or read/write in 4K byte increments to simulate the internal (EP)ROM of the 80C196 or external program memory.


intel

*PC XT, AT are trademarks of IBM.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel.

TRACE BUFFER

The ICE-196PC contains a 2K (2048) entry trace buffer for keeping a history of actual instruction execution. The trace buffer can be displayed as disassembled instructions or, optionally, disassembled instructions and the original C-96 and PL/M-96 source code.

BREAK SPECIFICATION

Three execution address breakpoints or one range of addresses can be active at any time. The ICE-196PC allows any number of breakpoints to be defined and activated when needed.

SYMBOLIC SUPPORT AND SOURCE CODE DISPLAY

Full ASM-96, PL/M-96 and C-96 language symbolics, including variable typing and scope, are supported by the ICE-196PC memory accesses, trace buffer display, breakpoint specification, and assembler/disassembler. Additionally, C-96 and PL/M-96 source code can be displayed to make development and debug easier.

SPECIFICATIONS

HOST REQUIREMENTS

IBM PC XT, AT (or compatible)
512K bytes RAM, Hard Disk
PC-DOS 3.0 or Later
One Unused Peripheral Slot
DC Current 2.5A

TARGET INTERFACE BOARD

Length 2.0" (5.1cm)
Height 1.2" (3.0cm)
Width 2.3" (5.8cm)

USER CABLE

Length 15.6" (39.6cm)

PROBE ELECTRICAL

80C196* plus per pin	50pf loading
	5ns propagation delay
Icc (from target system)	50mA @ 12 MHz
Operating Frequency	3.5 to 12 MHz, 12 MHz only with CPA

ENVIRONMENTAL CHARACTERISTICS

Operating Temperature	10°C to 40°C
	37.5°F to 104°F
Operating Humidity	Maximum 55% Relative Humidity, non-condensing

*This emulator supports the initial 80C196 microcontroller. The HOLD/HOLDA feature will be supported by a future product.

STANDALONE OPERATION

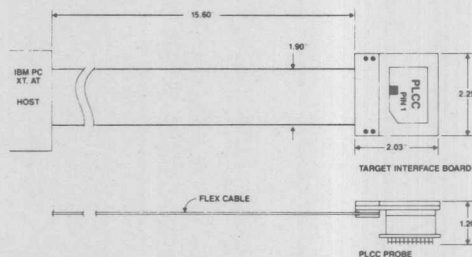
Product software can be developed prior to hardware availability with the optional Crystal Power Accessory (CPA) and the ICE-196PC mappable memory. The CPA also provides diagnostic testing to assure full functionality of the ICE-196PC.

VERSATILE AND POWERFUL HOST SOFTWARE

The ICE-196PC comes equipped with an on-line help facility, a dynamic command entry and syntax guide, built-in editor, assembler and disassembler, and the ability to customize the command set via literal definitions and debug procedures.

HOSTING

The ICE-196PC is hosted on the IBM PC XT, AT or compatibles with PC-DOS 3.0 or later.



ORDERING INFORMATION

Order Code	Description
ICE-196PC	Emulation Board, user cable, target interface board (PLCC), host, diagnostic, and tutorial software on 5 1/4" DOS diskette, and Crystal Power Accessory with power cable
ICE-196PCB	Same as above except does not include Crystal Power Accessory
CPA196	Crystal Power Accessory and power cable only
D86C96NL	C-96 Compiler*
D86PLM96NL	PL/M-96 Compiler*
D86ASM96NL	ASM-96 Assembler*

*Includes: Relocator/Linker, Object-to-hex Converter, Floating Point Arithmetic Library, Librarian

For more information or the number of your nearest sales office call 800-548-4725 (good in the U.S. and Canada).

UNITED STATES, Intel Corporation
3065 Bowers Ave., Santa Clara, CA 95051
Tel: (408) 987-8080

20

Additional Information

Digital circuits are often thought of as being immune to noise problems, but really they're not. Noises in digital systems produce software upsets: program jumps to apparently random locations in memory. Noise-induced glitches in the signal lines can cause such problems, but the supply voltage is more sensitive to glitches than the signal lines.

Severe noise conditions, those involving electrostatic discharges, or as found in automotive environments, can do permanent damage to the hardware. Electrostatic discharges can blow a crater in the silicon. In the automotive environment, in ordinary operation, the "12V" power line can show + and -400V transients.

This Application Note describes some electrical noises and noise environments. Design considerations, along the lines of PCB layout, power supply distribution and decoupling, and shielding and grounding techniques, that may help minimize noise susceptibility are reviewed. Special attention is given to the automotive and ESD environments.

Symptoms of Noise Problems

Noise problems are not usually encountered during the development phase of a microcontroller system. This is because benches rarely simulate the system's intended environment. Noise problems tend not to show up until the system is installed and operating in its intended environment. Then, after a few minutes or hours of normal operation the system finds itself somewhere out in left field. Inputs are ignored and outputs are gibberish. The system may respond to a reset, or it may have to be turned off physically and then back on again, at which point it commences operating as though nothing had happened. There may be an obvious cause, such as an electrostatic discharge from somebody's finger to a keyboard or the upset occurs every time a copier machine is turned on or off. Or there may be no obvious cause, and nothing the operator can do will make the upset repeat itself. But a few minutes, or a few hours, or a few days later it happens again.

One symptom of electrical noise problems is randomness, both in the occurrence of the problem and in what the system does in its failure. All operational upsets that occur at seemingly random intervals are not necessarily caused by noise in the system. Marginal VCC, inadequate decoupling, rarely encountered software conditions, or timing coincidences can produce upsets that seem to occur randomly. On the other hand, some noise sources can produce upsets downright periodically. Nevertheless, the more difficult it is to characterize an upset as to cause and effect, the more likely it is to be a noise problem.

Types and Sources of Electrical Noise

The name given to electrical noises other than those that are inherent in the circuit components (such as thermal noise) is EMI: electromagnetic interference. Motors, power switches, fluorescent lights, electrostatic discharges, etc., are sources of EMI. There is a veritable alphabet soup of EMI types, and these are briefly described below.

SUPPLY LINE TRANSIENTS

Anything that switches heavy current loads onto or off of AC or DC power lines will cause large transients in these power lines. Switching an electric typewriter on or off, for example, can put a 1000V spike onto the AC power lines.

The basic mechanism behind supply line transients is shown in Figure 1. The battery represents any power source, AC or DC. The coils represent the line inductance between the power source and the switchable loads R1 and R2. If both loads are drawing current, the line current flowing through the line inductance establishes a magnetic field of some value. Then, when one of the loads is switched off, the field due to that component of the line current collapses, generating transient voltages, $v = L(di/dt)$, which try to maintain the current at its original level. That's called an "inductive kick." Because of contact bounce, transients are generated whether the switch is being opened or closed, but they're worse when the switch is being opened.

An inductive kick of one type or another is involved in most line transients, including those found in the automotive environment. Other mechanisms for line transients exist, involving noise pickup on the lines. The noise voltages are then conducted to a susceptible circuit right along with the power.

EMP AND RFI

Anything that produces arcs or sparks will radiate electromagnetic pulses (EMP) or radio-frequency interference (RFI).

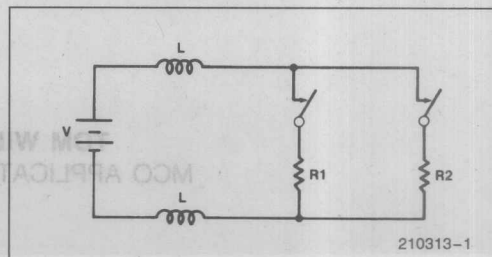


Figure 1. Supply Line Transients

Spark discharges have probably caused more software upsets in digital equipment than any other single noise source. The upsetting mechanism is the EMP produced by the spark. The EMP induces transients in the circuit, which are what actually cause the upset.

Arcs and sparks occur in automotive ignition systems, electric motors, switches, static discharges, etc. Electric motors that have commutator bars produce an arc as the brushes pass from one bar to the next. DC motors and the "universal" (AC/DC) motors that are used to power hand tools are the kinds that have commutator bars. In switches, the same inductive kick that puts transients on the supply lines will cause an opening or closing switch to throw a spark.

ESD

Electrostatic discharge (ESD) is the spark that occurs when a person picks up a static charge from walking across a carpet, and then discharges it into a keyboard, or whatever else can be touched. Walking across a carpet in a dry climate, a person can accumulate a static voltage of 35kV. The current pulse from an electrostatic discharge has an extremely fast risetime — typically, 4A/ns. Figure 2 shows ESD waveforms that have been observed by some investigators of ESD phenomena.

It is enlightening to calculate the $L(di/dt)$ voltage required to drive an ESD current pulse through a couple of inches of straight wire. Two inches of straight wire has about 50 nH of inductance. That's not very much, but using 50 nH for L and 4A/ns for di/dt gives an $L(di/dt)$ drop of about 200V. Recent observations by W.M. King suggest even faster risetimes (Figure 2b) and the occurrence of multiple discharges during a single discharge event.

Obviously, ESD-sensitivity needs to be considered in the design of equipment that is going to be subjected to it, such as office equipment.

GROUND NOISE

Currents in ground lines are another source of noise. These can be 60 Hz currents from the power lines, or RF hash, or crosstalk from other signals that are sharing this particular wire as a signal return line. Noise in the ground lines is often referred to as a "ground loop" problem. The basic concept of the ground loop is shown in Figure 3. The problem is that true earth-ground is not really at the same potential in all locations. If the two ends of a wire are earth-grounded at different locations, the voltage difference between the two "ground" points can drive significant currents (several amperes) through the wire. Consider the wire to be part of a loop which contains, in addition to the wire, a voltage source that represents the difference in potential between the two ground points, and you have

the classical "ground loop." By extension, the term is used to refer to any unwanted (and often unexpected) currents in a ground line.

"Radiated" and "Conducted" Noise

Radiated noise is noise that arrives at the victim circuit in the form of electromagnetic radiation, such as EMP and RFI. It causes trouble by inducing extraneous voltages in the circuit. Conducted noise is noise that arrives at the victim circuit already in the form of an extraneous voltage, typically via the AC or DC power lines.

One defends against radiated noise by care in designing layouts and the use of effective shielding techniques. One defends against conducted noise with filters and

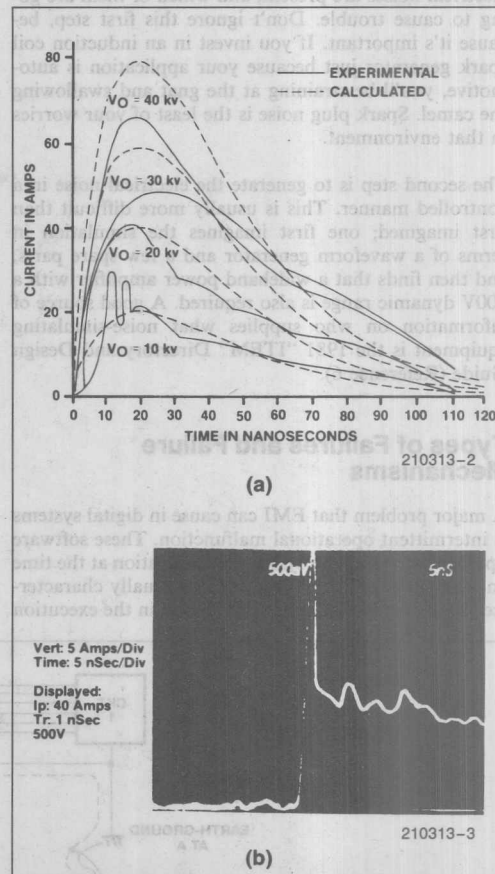


Figure 2. Waveforms of Electrostatic Discharge Currents From a Hand-Held Metallic Object

suppressors, although layouts and grounding techniques are important here, too.

Simulating the Environment

Addressing noise problems after the design of a system has been completed is an expensive proposition. The ill will generated by failures in the field is not cheap either. It's cheaper in the long run to invest a little time and money in learning about noise and noise simulation equipment, so that controlled tests can be made on the bench as the design is developing.

Simulating the intended noise environment is a two-step process: First you have to recognize what the noise environment is, that is, you have to know what kinds of electrical noises are present, and which of them are going to cause trouble. Don't ignore this first step, because it's important. If you invest in an induction coil spark generator just because your application is automotive, you'll be straining at the gnat and swallowing the camel. Spark plug noise is the least of your worries in that environment.

The second step is to generate the electrical noise in a controlled manner. This is usually more difficult than first imagined; one first imagines the simulation in terms of a waveform generator and a few spare parts, and then finds that a wideband power amplifier with a 200V dynamic range is also required. A good source of information on who supplies what noise-simulating equipment is the 1981 "ITEM" Directory and Design Guide (Reference 6).

Types of Failures and Failure Mechanisms

A major problem that EMI can cause in digital systems is intermittent operational malfunction. These software upsets occur when the system is in operation at the time an EMI source is activated, and are usually characterized by a loss of information or a jump in the execution

of the program to some random location in memory. The person who has to iron out such problems is tempted to say the program counter went crazy. There is usually no damage to the hardware, and normal operation can resume as soon as the EMI has passed or the source is de-activated. Resuming normal operation usually requires manual or automatic reset, and possibly re-entering of lost information.

Electrostatic discharges from operating personnel can cause not only software upsets, but also permanent ("hard") damage to the system. For this to happen the system doesn't even have to be in operation. Sometimes the permanent damage is latent, meaning the initial damage may be marginal and require further aggravation through operating stress and time before permanent failure takes place. Sometimes too the damage is hidden.

One ESD-related failure mechanism that has been identified has to do with the bias voltage on the substrate of the chip. On some CPU chips the substrate is held at $-2.5V$ by a phase-shift oscillator working into a capacitor/diode clamping circuit. This is called a "charge pump" in chip-design circles. If the substrate wanders too far in either direction, program read errors are noted. Some designs have been known to allow electrostatic discharge currents to flow directly into port pins of an 8048. The resulting damage to the oxide causes an increase in leakage current, which loads down the charge pump, reducing the substrate voltage to a marginal or unacceptable level. The system is then unreliable or completely inoperative until the CPU chip is replaced. But if the CPU chip was subjected to a discharge spark once, it will eventually happen again.

Chips that have a grounded substrate, such as the 8748, can sometimes sustain some oxide damage without actually becoming inoperative. In this case the damage is present, and the increased leakage current is noted; however, since the substrate voltage retains its design value, the damage is largely hidden.

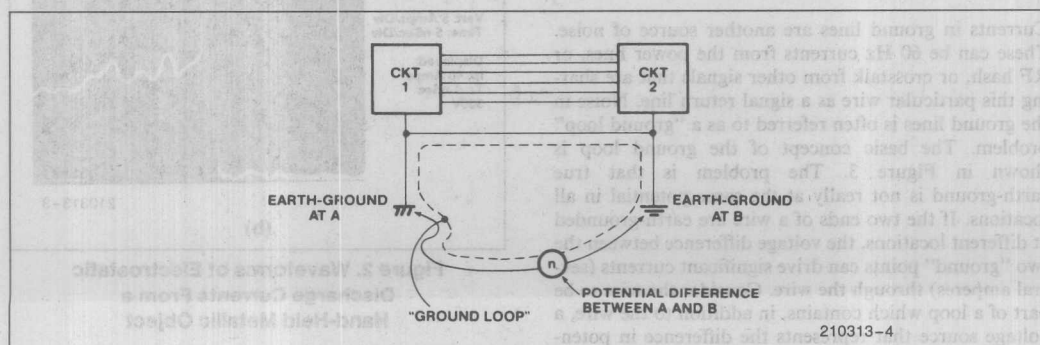


Figure 3. What a Ground Loop Is

It must therefore be recognized that connecting port pins unprotected to a keyboard or to anything else that is subject to electrostatic discharges, makes an extremely dangerous configuration. It doesn't make any difference what CPU chip is being used, or who makes it. If it connects unprotected to a keyboard, it will eventually be destroyed. Designing for an ESD-environment will be discussed further on.

We might note here that MOS chips are not the only components that are susceptible to permanent ESD damage. Bipolar and linear chips can also be damaged in this way. PN junctions are subject to a hard failure mechanism called thermal secondary breakdown, in which a current spike, such as from an electrostatic discharge, causes microscopically localized spots in the junction to approach melt temperatures. Low power TTL chips are subject to this type of damage, as are op-amps. Op-amps, in addition, often carry on-chip MOS capacitors which are directly across an external pin combination, and these are susceptible to dielectric breakdown.

We return now to the subject of software upsets. Noise transients can upset the chip through any pin, even an output pin, because every pin on the chip connects to the substrate through a pn junction. However, the most vulnerable pin is probably the VCC line, since it has direct access to all parts of the chip: every register, gate, flip-flop and buffer.

The menu of possible upset mechanisms is quite lengthy. A transient on the substrate at the wrong time will generally cause a program read error. A false level at a control input can cause an extraneous or misdirected opcode fetch. A disturbance on the supply line can flip a bit in the program counter or instruction register. A short interruption or reversal of polarity on the supply line can actually turn the processor off, but not long enough for the power-up reset capacitor to discharge. Thus when the transient ends, the chip starts up again without a reset.

A common failure mode is for the processor to lock itself into a tight loop. Here it may be executing the data in a table, or the program counter may have jumped a notch, so that the processor is now executing operands instead of opcodes, or it may be trying to fetch opcodes from a nonexistent external program memory.

It should be emphasized that mechanisms for upsets have to do with the arrival of noise-induced transients at the pins of the chips, rather than with the generation of noise pulses within the chip itself, that is, it's not the chip that is picking up noise, it's the circuit.

The Game Plan

Prevention is usually cheaper than suppression, so first we'll consider some preventive methods that might help

to minimize the generation of noise voltages in the circuit. These methods involve grounding, shielding, and wiring techniques that are directed toward the mechanisms by which noise voltages are generated in the circuit. We'll also discuss methods of decoupling. Then we'll look at some schemes for making a graceful recovery from upsets that occur in spite of preventive measures. Lastly, we'll take another look at two special problem areas: electrostatic discharges and the automotive environment.

Current Loops

The first thing most people learn about electricity is that current won't flow unless it can flow in a closed loop. This simple fact is sometimes temporarily forgotten by the overworked engineer who has spent the past several years mastering the intricacies of the DO loop, the timing loop, the feedback loop, and maybe even the ground loop. The simple current loop probably owes its apparent demise to the invention of the ground symbol. By a stroke of the pen one avoids having to draw the return paths of most of the current loops in the circuit. Then "ground" turns into an infinite current sink, so that any current that flows into it is gone and forgotten. Forgotten it may be, but it's not gone. It must return to its source, so that its path will by all the laws of nature form a closed loop.

The physical geometry of a given current loop is the key to why it generates EMI, why it's susceptible to EMI, and how to shield it. Specifically, it's the area of the loop that matters.

Any flow of current generates a magnetic field whose intensity varies inversely to the distance from the wire that carries the current. Two parallel wires conducting currents $+I$ and $-I$ (as in signal feed and return lines) would generate a nonzero magnetic field near the wires, where the distance from a given point to one wire is noticeably different from the distance to the other wire, but farther away (relative to the wire spacing), where the distances from a given point to either wire are about the same, the fields from both wires tend to cancel out. Thus, maintaining proximity between feed and return paths is an important way to minimize their interference with other signals. The way to maintain their proximity is essentially to minimize their loop area. And, because the mutual inductance from current loop A to current loop B is the same as the mutual inductance from current loop B to current loop A, a circuit that doesn't radiate interference doesn't receive it either.

Thus, from the standpoint of reducing both generation of EMI and susceptibility to EMI, the hard rule is to keep loop areas small. To say that loop areas should be minimized is the same as saying the circuit inductance

should be minimized. Inductance is by definition the constant of proportionality between current and the magnetic field it produces: $\phi = LI$. Holding the feed and return wires close together so as to promote field cancellation can be described either as minimizing the loop area or as minimizing L . It's the same thing.

Shielding

There are three basic kinds of shields: shielding against capacitive coupling, shielding against inductive coupling, and RF shielding. Capacitive coupling is electric field coupling, so shielding against it amounts to shielding against electric fields. As will be seen, this is relatively easy. Inductive coupling is magnetic field coupling, so shielding against it is shielding against magnetic fields. This is a little more difficult. Strangely enough, this type of shielding does not in general involve the use of magnetic materials. RF shielding, the classical "metallic barrier" against all sorts of electromagnetic fields, is what most people picture when they think about shielding. Its effectiveness depends partly on the selection of the shielding material, but mostly, as it turns out, on the treatment of its seams and the geometry of its openings.

SHIELDING AGAINST CAPACITIVE COUPLING

Capacitive coupling involves the passage of interfering signals through mutual or stray capacitances that aren't shown on the circuit diagram, but which the experienced engineer knows are there. Capacitive coupling to one's body is what would cause an unstable oscillator to change its frequency when the person reaches his hand over the circuit, for example. More importantly, in a digital system it causes crosstalk in multi-wire cables.

The way to block capacitive coupling is to enclose the circuit or conductor you want to protect in a metal shield. That's called an electrostatic or Faraday shield. If coverage is 100%, the shield does not have to be grounded, but it usually is, to ensure that circuit-to-shield capacitances go to signal reference ground rather than act as feedback and crosstalk elements. Besides, from a mechanical point of view, grounding it is almost inevitable.

A grounded Faraday shield can be used to break capacitive coupling between a noisy circuit and a victim circuit, as shown in Figure 4. Figure 4a shows two circuits capacitively coupled through the stray capacitance between them. In Figure 4b the stray capacitance is intercepted by a grounded Faraday shield, so that interference currents are shunted to ground. For example, a grounded plane can be inserted between PCBs (printed circuit boards) to eliminate most of the capacitive coupling between them.

Another application of the Faraday shield is in the electrostatically shielded transformer. Here, a conducting foil is laid between the primary and secondary coils so as to intercept the capacitive coupling between them. If a system is being upset by AC line transients, this type of transformer may provide the fix. To be effective in this application, the shield must be connected to the greenwire ground.

SHIELDING AGAINST INDUCTIVE COUPLING

With inductive coupling, the physical mechanism involved is a magnetic flux density B from some external interference source that links with a current loop in the victim circuit, and generates a voltage in the loop in accordance with Lenz's law: $v = -NA(dB/dt)$, where in this case $N = 1$ and A is the area of the current loop in the victim circuit.

There are two aspects to defending a circuit against inductive pickup. One aspect is to try to minimize the offensive fields at their source. This is done by minimizing the area of the current loop at the source so as to promote field cancellation, as described in the section on current loops. The other aspect is to minimize the inductive pickup in the victim circuit by minimizing the area of that current loop, since, from Lenz's law, the induced voltage is proportional to this area. So the two aspects really involve the same corrective action: minimize the areas of the current loops. In other words, minimizing the offensiveness of a circuit inherently minimizes its susceptibility.

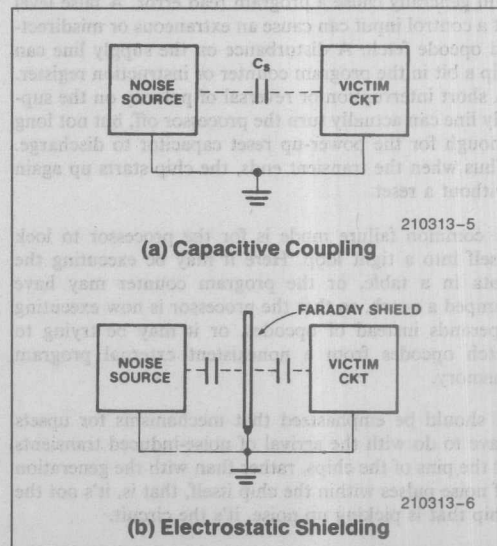


Figure 4. Use of Faraday Shield

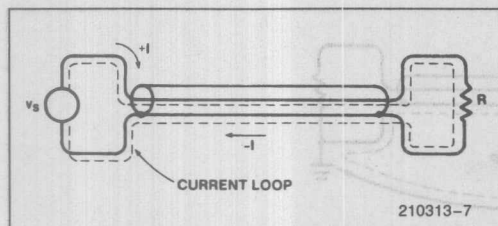


Figure 5. External to the Shield, $\phi = 0$

Shielding against inductive coupling means nothing more nor less than controlling the dimensions of the current loops in the circuit. We must look at four examples of this type of "shielding": the coaxial cable, the twisted pair, the ground plane, and the gridded-ground PCB layout.

The Coaxial Cable—Figure 5 shows a coaxial cable carrying a current I from a signal source to a receiving load. The shield carries the same current as the center conductor. Outside the shield, the magnetic field produced by $+I$ flowing in the center conductor is cancelled by the field produced by $-I$ flowing in the shield. To the extent that the cable is ideal in producing zero external magnetic field, it is immune to inductive pickup from external sources. The cable adds effectively zero area to the loop. This is true only if the shield carries the same current as the center conductor.

In the real world, both the signal source and the receiving load are likely to have one end connected to a common signal ground. In that case, should the cable be grounded at one end, both ends, or neither end? The answer is that it should be grounded at both ends. Figure 6a shows the situation when the cable shield is grounded at only one end. In that case the current loop runs down the center conductor of the cable, then back through the common ground connection. The loop area is not well defined. The shield not only does not carry the same current as the center conductor, but it doesn't carry any current at all. There is no field cancellation at all. The shield has no effect whatsoever on either the generation of EMI or susceptibility to EMI. (It is, however, still effective as an electrostatic shield, or at least it would be if the shield coverage were 100%.)

Figure 6b shows the situation when the cable is grounded at both ends. Does the shield carry all of the return current, or only a portion of it on account of the shunting effect of the common ground connection? The answer to that question depends on the frequency content of the signal. In general, the current loop will follow the path of least impedance. At low frequencies, 0 Hz to several kHz, where the inductive reactance is insignificant, the current will follow the path of least resistance. Above a few kHz, where inductive reactance predominates, the current will follow the path of least inductance. The path of least inductance is the path of

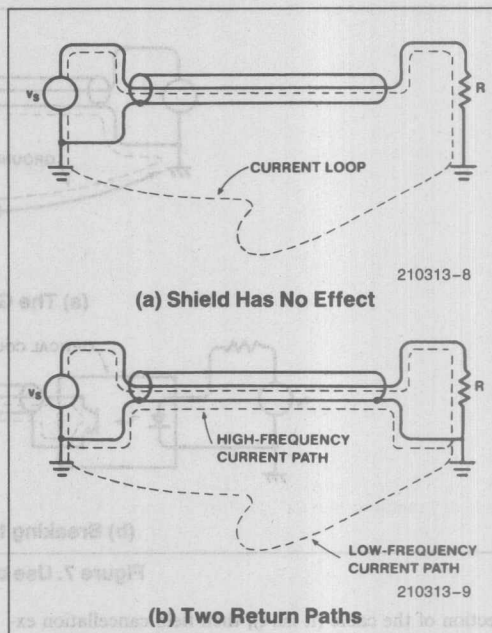


Figure 6. Use of Coaxial Cable

minimum loop area. Hence, for higher frequencies the shield carries virtually the same current as the center conductor, and is therefore effective against both generation and reception of EMI.

Note that we have now introduced the famous "ground loop" problem, as shown in Figure 7a. Fortunately, a digital system has some built-in immunity to moderate ground loop noise. In a noisy environment, however, one can break the ground loop, and still maintain the shielding effectiveness of the coaxial cable, by inserting an optical coupler, as shown in Figure 7b. What the optical coupler does, basically, is allow us to re-define the signal source as being ungrounded, so that that end of the cable need not be grounded, and still lets the shield carry the same current as the center conductor. Obviously, if the signal source weren't grounded in the first place, the optical coupler wouldn't be needed.

The Twisted Pair—A cheaper way to minimize loop area is to run the feed and return wires right next to each other. This isn't as effective as a coaxial cable in minimizing loop area. An ideal coaxial cable adds zero area to the loop, whereas merely keeping the feed and return wires next to each other is bound to add a finite area.

However, two things work to make this cheaper method almost as good as a coaxial cable. First, real coaxial cables are not ideal. If the shield current isn't evenly distributed around the center conductor at every cross-

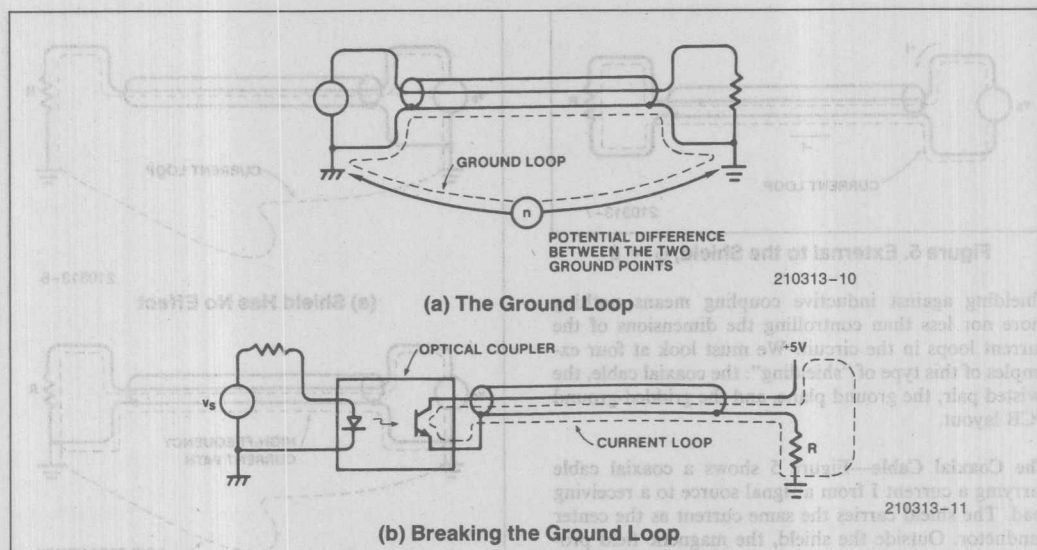


Figure 7. Use of Optical Coupler

section of the cable (it isn't), then field cancellation external to the shield is incomplete. If field cancellation is incomplete, then the effective area added to the loop by the cable isn't zero. Second, in the cheaper method the feed and return wires can be twisted together. This not only maintains their proximity, but the noise picked up in one twist tends to cancel out the noise picked up in the next twist down the line. Thus the "twisted pair" turns out to be about as good a shield against inductive coupling as coaxial cable is.

The twisted pair does not, however, provide electrostatic shielding (i.e., shielding against capacitive coupling). Another operational difference between them is that the coaxial cable works better at higher frequencies. This is primarily because the twisted pair adds more capacitive loading to the signal source than the coaxial cable does. The twisted pair is normally considered useful up to only about 1 MHz, as opposed to near a GHz for the coaxial cable.

The Ground Plane—The best way to minimize loop areas when many current loops are involved is to use a ground plane. A ground plane is a conducting surface that is to serve as a return conductor for all the current loops in the circuit. Normally, it would be one or more layers of a multilayer PCB. All ground points in the circuit go not to a grounded trace on the PCB, but directly to the ground plane. This leaves each current loop in the circuit free to complete itself in whatever configuration yields minimum loop area (for frequencies wherein the ground path impedance is primarily inductive).

Thus, if the feed path for a given signal zigzags its way across the PCB, the return path for this signal is free to zigzag right along beneath it on the ground plane, in such a configuration as to minimize the energy stored in the magnetic field produced by this current loop. Minimal magnetic flux means minimal effective loop area and minimal susceptibility to inductive coupling.

The Gridded-Ground PCB Layout—The next best thing to a ground plane is to lay out the ground traces on a PCB in the form of a grid structure, as shown in Figure 8. Laying horizontal traces on one side of the board and vertical traces on the other side allows the passage of signal and power traces. Wherever vertical and horizontal ground traces cross, they must be connected by a feed-through.

Have we not created here a network of "ground loops"? Yes, in the literal sense of the word, but loops in the ground layout on a PCB are not to be feared. Such inoffensive little loops have never caused as much noise pickup as their avoidance has. Trying to avoid innocent little loops in the ground layout, PCB designers have forced current loops into geometries that could swallow a whale. That is exactly the wrong thing to do.

The gridded ground structure works almost as well as the ground plane, as far as minimizing loop area is concerned. For a given current loop, the primary return path may have to zig once in a while where its feed path zags, but you still get a mathematically optimal dis-

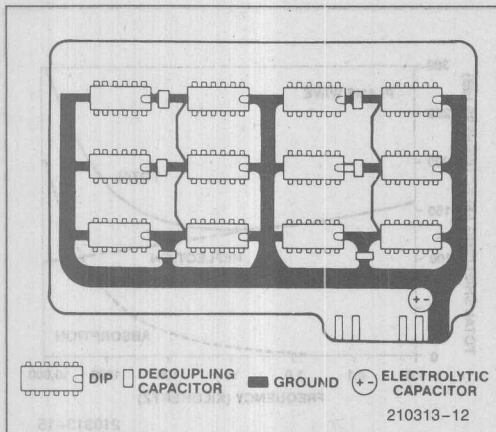


Figure 8. PCB with Gridded Ground

tribution of currents in the grid structure, such that the current loop produces less magnetic flux than if the return path were restrained to follow any single given ground trace. The key to attaining minimum loop areas for all the current loops together is to let the ground currents distribute themselves around the entire area of the board as freely as possible. They want to minimize their own magnetic field. Just let them.

RF SHIELDING

A time-varying electric field generates a time-varying magnetic field, and vice versa. Far from the source of a time-varying EM field, the ratio of the amplitudes of the electric and magnetic fields is always 377Ω . Up close to the source of the fields, however, this ratio can be quite different, and dependent on the nature of the source. Where the ratio is near 377Ω is called the far field, and where the ratio is significantly different from 377Ω is called the near field. The ratio itself is called the wave impedance, E/H .

The near field goes out about $1/6$ of a wavelength from the source. At 1 MHz this is about 150 feet, and at 10 MHz it's about 15 feet. That means if an EMI source is in the same room with the victim circuit, it's likely to be a near field problem. The reason this matters is that in the near field an RF interference problem could be almost entirely due to E-field coupling or H-field coupling, and that could influence the choice of an RF shield or whether an RF shield will help at all.

In the near field of a whip antenna, the E/H ratio is higher than 377Ω , which means it's mainly an E-field generator. A wire-wrap post can be a whip antenna. Interference from a whip antenna would be by electric field coupling, which is basically capacitive coupling. Methods to protect a circuit from capacitive coupling, such as a Faraday shield, would be effective

against RF interference from a whip antenna. A gridded-ground structure would be less effective.

In the near field of a loop antenna, the E/H ratio is lower than 377Ω , which means it's mainly an H-field generator. Any current loop is a loop antenna. Interference from a loop antenna would be by magnetic field coupling, which is basically the same as inductive coupling. Methods to protect a circuit from inductive coupling, such as a gridded-ground structure, would be effective against RF interference from a loop antenna. A Faraday shield would be less effective.

A more difficult case of RF interference, near field or far field, may require a genuine metallic RF shield. The idea behind RF shielding is that time-varying EMI fields induce currents in the shielding material. The induced currents dissipate energy in two ways: I^2R losses in the shielding material and radiation losses as they re-radiate their own EM fields. The energy for both of these mechanisms is drawn from the impinging EMI fields. Hence the EMI is weakened as it penetrates the shield.

More formally, the I^2R losses are referred to as absorption loss, and the re-radiation is called reflection loss. As it turns out, absorption loss is the primary shielding mechanism for H-fields, and reflection loss is the primary shielding mechanism for E-fields. Reflection loss, being a surface phenomenon, is pretty much independent of the thickness of the shielding material. Both loss mechanisms, however, are dependent on the frequency (ω) of the impinging EMI field, and on the permeability (μ) and conductivity (σ) of the shielding material. These loss mechanisms vary approximately as follows:

$$\text{reflection loss to an E-field (in dB)} \sim \log \frac{\sigma}{\omega \mu}$$

$$\text{absorption loss to an H-field (in dB)} \sim t \sqrt{\omega \sigma \mu}$$

where t is the thickness of the shielding material.

The first expression indicates that E-field shielding is more effective if the shield material is highly conductive, and less effective if the shield is ferromagnetic, and that low-frequency fields are easier to block than high-frequency fields. This is shown in Figure 9.

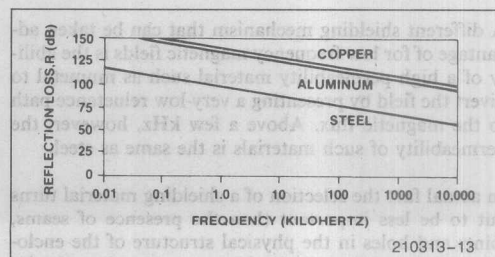


Figure 9. E-Field Shielding

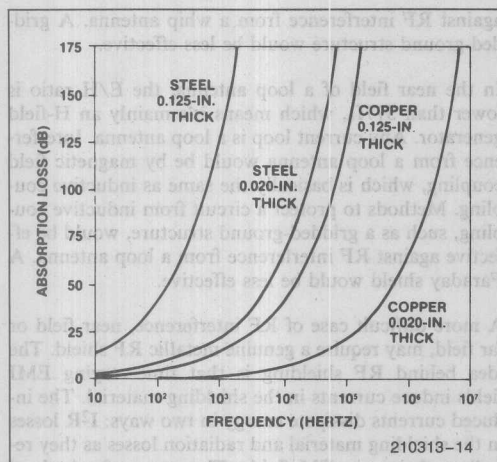


Figure 10. H-Field Shielding

Copper and aluminum both have the same permeability, but copper is slightly more conductive, and so provides slightly greater reflection loss to an E-field. Steel is less effective for two reasons. First, it has a somewhat elevated permeability due to its iron content, and second, as tends to be the case with magnetic materials, it is less conductive.

On the other hand, according to the expression for absorption loss to an H-field, H-field shielding is more effective at higher frequencies and with shield material that has both high conductivity and high permeability. In practice, however, selecting steel for its high permeability involves some compromise in conductivity. But the increase in permeability more than makes up for the decrease in conductivity, as can be seen in Figure 10. This figure also shows the effect of shield thickness.

A composite of E-field and H-field shielding is shown in Figure 11. However, this type of data is meaningful only in the far field. In the near field the EMI could be 90% H-field, in which case the reflection loss is irrelevant. It would be advisable then to beef up the absorption loss, at the expense of reflection loss, by choosing steel. A better conductor than steel might be less expensive, but quite ineffective.

A different shielding mechanism that can be taken advantage of for low frequency magnetic fields is the ability of a high permeability material such as mumetal to divert the field by presenting a very low reluctance path to the magnetic flux. Above a few kHz, however, the permeability of such materials is the same as steel.

In actual fact the selection of a shielding material turns out to be less important than the presence of seams, joints and holes in the physical structure of the enclosure. The shielding mechanisms are related to the induction of currents in the shield material, but the cur-

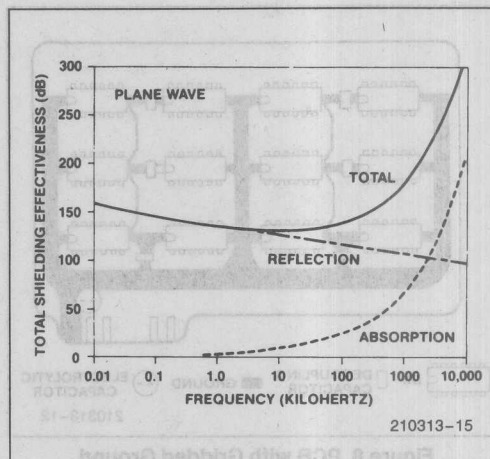


Figure 11. E- and H-Field Shielding

rents must be allowed to flow freely. If they have to detour around slots and holes, as shown in Figure 12, the shield loses much of its effectiveness.

As can be seen in Figure 12, the severity of the detour has less to do with the area of the hole than it does with the geometry of the hole. Comparing Figure 12c with 12d shows that a long narrow discontinuity such as a seam can cause more RF leakage than a line of holes with larger total area. A person who is responsible for designing or selecting rack or chassis enclosures for an EMI environment needs to be familiar with the techniques that are available for maintaining electrical continuity across seams. Information on these techniques is available in the references.

Grounds

There are two kinds of grounds: earth-ground and signal ground. The earth is not an equipotential surface, so earth ground potential varies. That and its other electrical properties are not conducive to its use as a return conductor in a circuit. However, circuits are often connected to earth ground for protection against shock hazards. The other kind of ground, signal ground, is an arbitrarily selected reference node in a circuit—the node with respect to which other node voltages in the circuit are measured.

SAFETY GROUND

The standard 3-wire single-phase AC power distribution system is represented in Figure 13. The white wire is earth-grounded at the service entrance. If a load circuit has a metal enclosure or chassis, and if the black wire develops a short to the enclosure, there will be a shock hazard to operating personnel, unless the enclosure itself is earth-grounded. If the enclosure is earth-

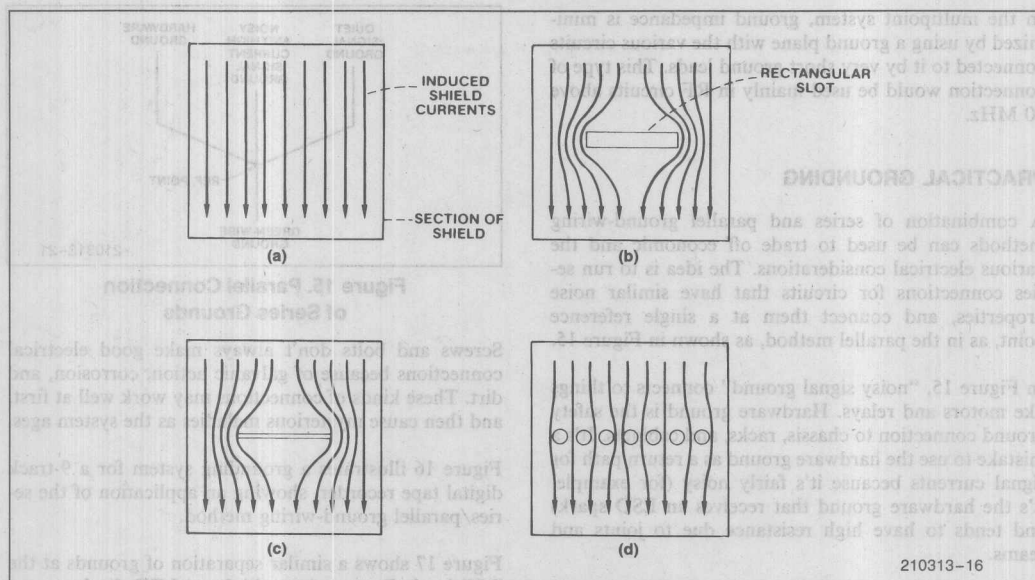


Figure 12. Effect of Shield Discontinuity on Magnetically Induced Shield Current

grounded, a short results in a blown fuse rather than a "hot" enclosure. The earth-ground connection to the enclosure is called a safety ground. The advantage of the 3-wire power system is that it distributes a safety ground along with the power.

Note that the safety-ground wire carries no current, except in case of a fault, so that at least for low frequencies it's at earth-ground potential along its entire length. The white wire, on the other hand, may be several volts off ground, due to the IR drop along its length.

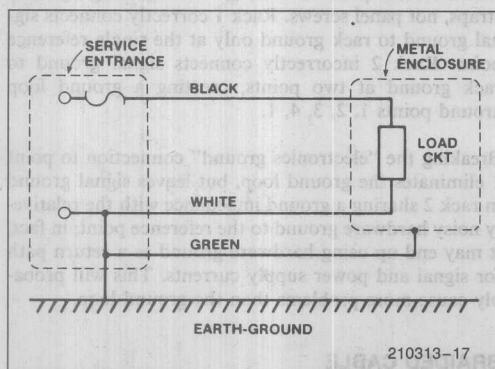


Figure 13. Single-Phase Power Distribution

SIGNAL GROUND

Signal ground is a single point in a circuit that is designated to be the reference node for the circuit. Commonly, wires that connect to this single point are also referred to as "signal ground." In some circles "power supply common" or PSC is the preferred terminology for these conductors. In any case, the manner in which these wires connect to the actual reference point is the basis of distinction among three kinds of signal-ground wiring methods: series, parallel, and multipoint. These methods are shown in Figure 14.

The series connection is pretty common because it's simple and economical. It's the noisiest of the three, however, due to common ground impedance coupling between the circuits. When several circuits share a ground wire, currents from one circuit, flowing through the finite impedance of the common ground line, cause variations in the ground potential of the other circuits. Given that the currents in a digital system tend to be spiked, and that the common impedance is mainly inductive reactance, the variations could be bad enough to cause bit errors in high current or particularly noisy situations.

The parallel connection eliminates common ground impedance problems, but uses a lot of wire. Other disadvantages are that the impedance of the individual ground lines can be very high, and the ground lines themselves can become sources of EMI.

In the multipoint system, ground impedance is minimized by using a ground plane with the various circuits connected to it by very short ground leads. This type of connection would be used mainly in RF circuits above 10 MHz.

PRACTICAL GROUNDING

A combination of series and parallel ground-wiring methods can be used to trade off economic and the various electrical considerations. The idea is to run series connections for circuits that have similar noise properties, and connect them at a single reference point, as in the parallel method, as shown in Figure 15.

In Figure 15, "noisy signal ground" connects to things like motors and relays. Hardware ground is the safety ground connection to chassis, racks, and cabinets. It's a mistake to use the hardware ground as a return path for signal currents because it's fairly noisy (for example, it's the hardware ground that receives an ESD spark) and tends to have high resistance due to joints and seams.

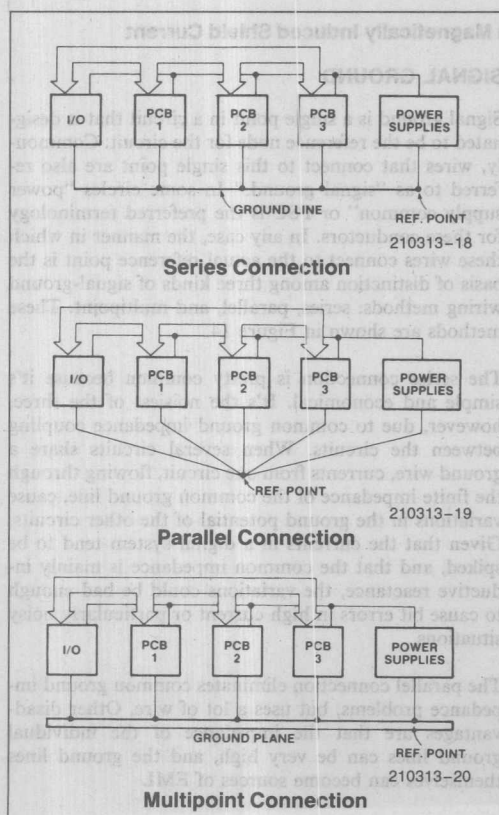


Figure 14. Three Ways to Wire the Grounds

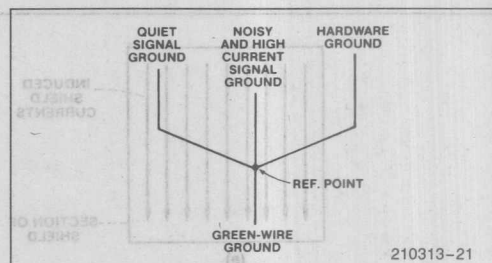


Figure 15. Parallel Connection of Series Grounds

Screws and bolts don't always make good electrical connections because of galvanic action, corrosion, and dirt. These kinds of connections may work well at first, and then cause mysterious maladies as the system ages.

Figure 16 illustrates a grounding system for a 9-track digital tape recorder, showing an application of the series/parallel ground-wiring method.

Figure 17 shows a similar separation of grounds at the PCB level. Currents in multiplexed LED displays tend to put a lot of noise on the ground and supply lines because of the constant switching and changing involved in the scanning process. The segment driver ground is relatively quiet, since it doesn't conduct the LED currents. The digit driver ground is noisier, and should be provided with a separate path to the PCB ground terminal, even if the PCB ground layout is gridded. The LED feed and return current paths should be laid out on opposite sides of the board like parallel flat conductors.

Figure 18 shows right and wrong ways to make ground connections in racks. Note that the safety ground connections from panel to rack are made through ground straps, not panel screws. Rack 1 correctly connects signal ground to rack ground only at the single reference point. Rack 2 incorrectly connects signal ground to rack ground at two points, creating a ground loop around points 1, 2, 3, 4, 1.

Breaking the "electronics ground" connection to point 1 eliminates the ground loop, but leaves signal ground in rack 2 sharing a ground impedance with the relatively noisy hardware ground to the reference point; in fact, it may end up using hardware ground as a return path for signal and power supply currents. This will probably cause more problems than the ground loop.

BRAIDED CABLE

Ground impedance problems can be virtually eliminated by using braided cable. The reduction in impedance is due to skin effect: At higher frequencies the current tends to flow along the surface of a conductor rather

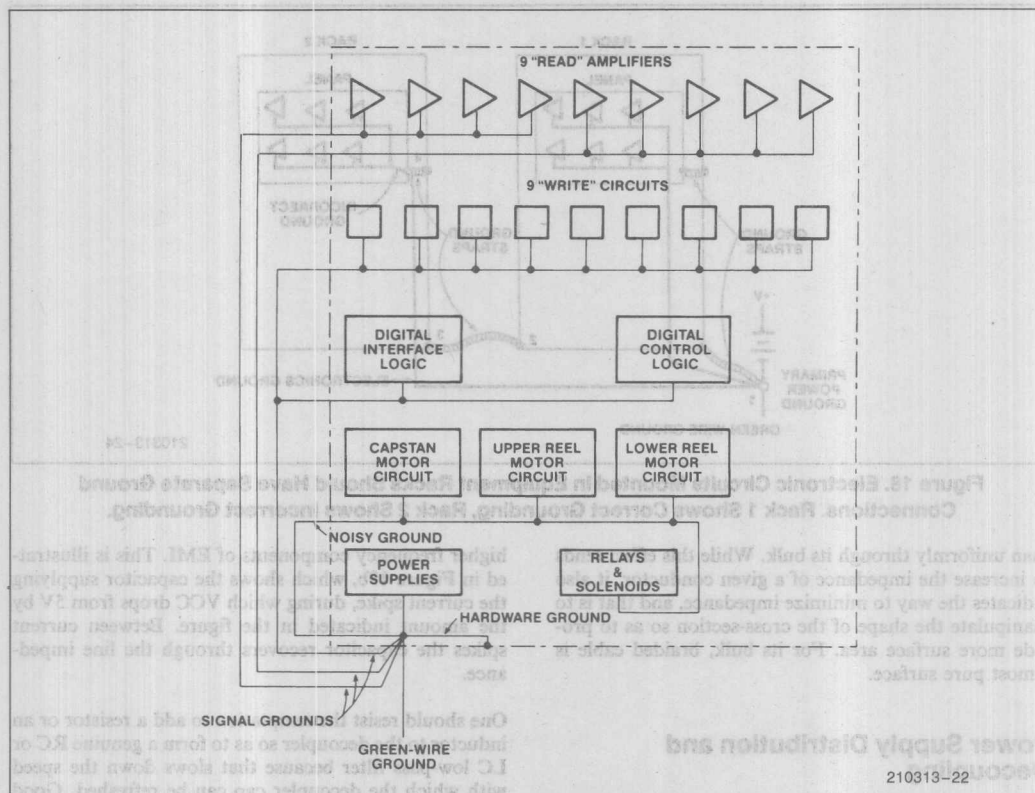


Figure 16. Ground System in a 9-Track Digital Recorder

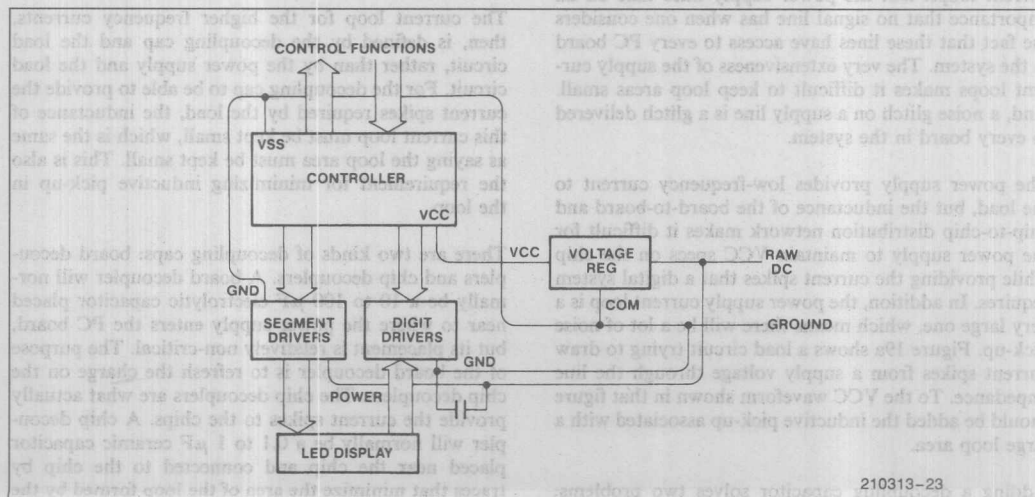


Figure 17. Separate Ground for Multiplexed LED Display

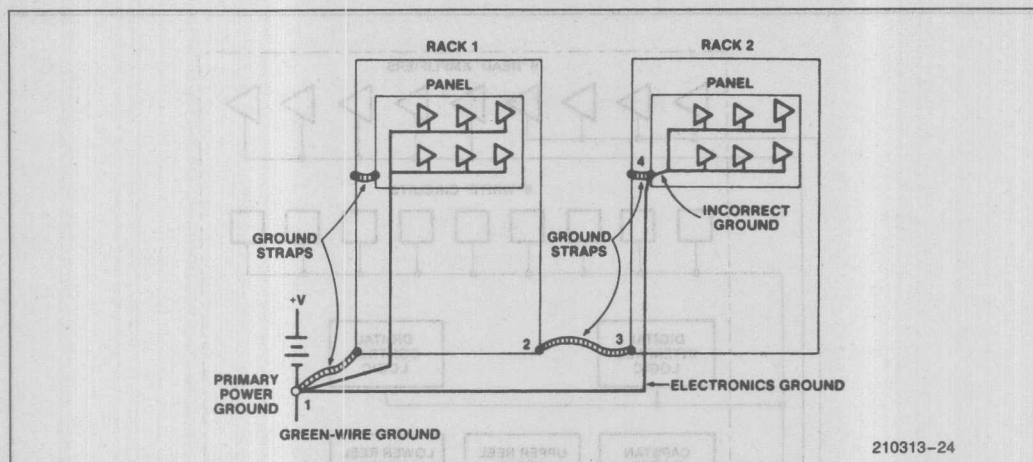


Figure 18. Electronic Circuits Mounted in Equipment Racks Should Have Separate Ground Connections. Rack 1 Shows Correct Grounding, Rack 2 Shows Incorrect Grounding.

than uniformly through its bulk. While this effect tends to increase the impedance of a given conductor, it also indicates the way to minimize impedance, and that is to manipulate the shape of the cross-section so as to provide more surface area. For its bulk, braided cable is almost pure surface.

Power Supply Distribution and Decoupling

The main consideration for power supply distribution lines is, as for signal lines, to minimize the areas of the current loops. But the power supply lines take on an importance that no signal line has when one considers the fact that these lines have access to every PC board in the system. The very extensiveness of the supply current loops makes it difficult to keep loop areas small. And, a noise glitch on a supply line is a glitch delivered to every board in the system.

The power supply provides low-frequency current to the load, but the inductance of the board-to-board and chip-to-chip distribution network makes it difficult for the power supply to maintain VCC specs on the chip while providing the current spikes that a digital system requires. In addition, the power supply current loop is a very large one, which means there will be a lot of noise pick-up. Figure 19a shows a load circuit trying to draw current spikes from a supply voltage through the line impedance. To the VCC waveform shown in that figure should be added the inductive pick-up associated with a large loop area.

Adding a decoupling capacitor solves two problems: The capacitor acts as a nearby source of charge to supply the current spikes through a smaller line impedance, and it defines a much smaller loop area for the

higher frequency components of EMI. This is illustrated in Figure 19b, which shows the capacitor supplying the current spike, during which VCC drops from 5V by the amount indicated in the figure. Between current spikes the capacitor recovers through the line impedance.

One should resist the temptation to add a resistor or an inductor to the decoupler so as to form a genuine RC or LC low-pass filter because that slows down the speed with which the decoupler cap can be refreshed. Good filtering and good decoupling are not necessarily the same thing.

The current loop for the higher frequency currents, then, is defined by the decoupling cap and the load circuit, rather than by the power supply and the load circuit. For the decoupling cap to be able to provide the current spikes required by the load, the inductance of this current loop must be kept small, which is the same as saying the loop area must be kept small. This is also the requirement for minimizing inductive pick-up in the loop.

There are two kinds of decoupling caps: board decouplers and chip decouplers. A board decoupler will normally be a 10 to 100 μF electrolytic capacitor placed near to where the power supply enters the PC board, but its placement is relatively non-critical. The purpose of the board decoupler is to refresh the charge on the chip decouplers. The chip decouplers are what actually provide the current spikes to the chips. A chip decoupler will normally be a 0.1 to 1 μF ceramic capacitor placed near the chip and connected to the chip by traces that minimize the area of the loop formed by the cap and the chip. If a chip decoupler is not properly placed on the board, it will be ineffective as a decoupler

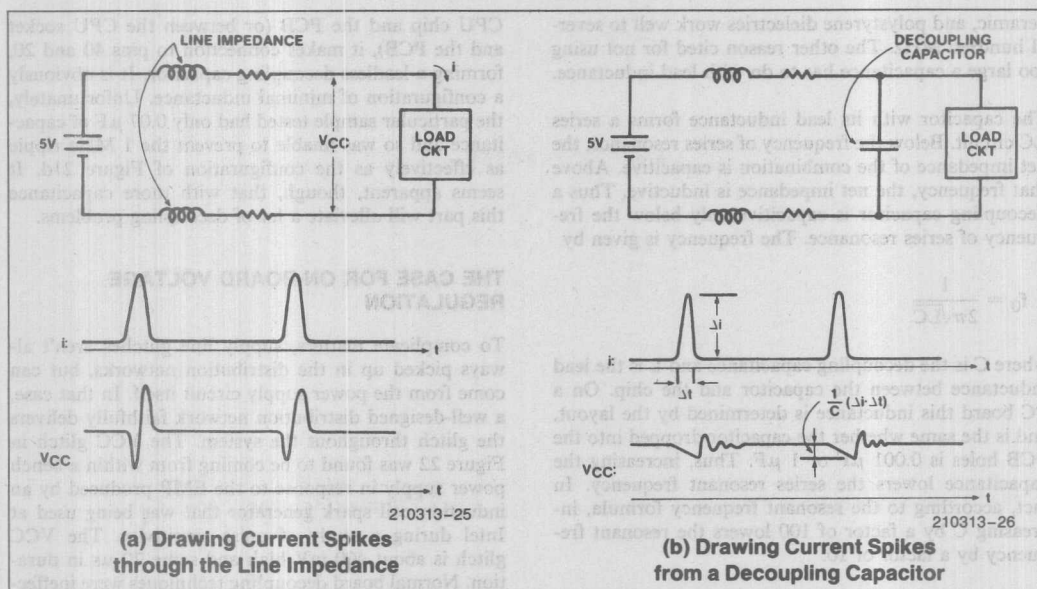


Figure 19. What a Decoupling Capacitor Does

and will serve only to increase the cost of the board. Good and bad placement of decoupling capacitors are illustrated in Figure 20.

Power distribution traces on the PC board need to be laid out so as to obtain minimal area (minimal inductance) in the loops formed by each chip and its decoupler, and by the chip decouplers and the board decoupler. One way to accomplish this goal is to use a power plane. A power plane is the same as a ground plane, but at VCC potential. More economically, a power grid similar to the ground grid previously discussed (Figure 8) can be used. Actually, if the chip decoupling loops are small, other aspects of the power layout are less critical. In other words, power planes and power gridding aren't needed, but power traces *should* be laid in the closest possible proximity to ground traces, prefer-

ably so that each power trace is on the direct opposite side of the board from a ground trace.

Special-purpose power supply distribution buses which mount on the PCB are available. The buses use a parallel flat conductor configuration, one conductor being a VCC line and the other a ground line. Used in conjunction with a gridded ground layout, they not only provide a low-inductance distribution system, but can themselves form part of the ground grid, thus facilitating the PCB layout. The buses are available with and without enhanced bus capacitance, under the names Mini/Bus® and Q/PAC® from Rogers Corp. (5750 E. McKellips, Mesa, AZ 85205).

SELECTING THE VALUE OF THE DECOUPLING CAP

The effectiveness of the decoupling cap has a lot to do with the way the power and ground traces connect this capacitor to the chip. In fact, the area formed by this loop is more important than the value of the capacitance. Then, given that the area of this loop is indeed minimal, it can generally be said that the larger the value of the decoupling cap, the more effective it is, if the cap has a mica, ceramic, glass, or polystyrene dielectric.

It's often said, and not altogether accurately, that the chip decoupler shouldn't have too large a value. There are two reasons for this statement. One is that some capacitors, because of the nature of their dielectrics, tend to become inductive or lossy at higher frequencies. This is true of electrolytic capacitors, but mica, glass,

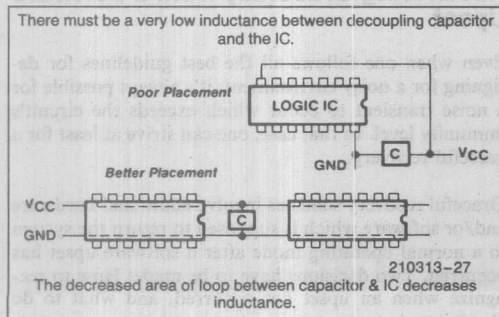


Figure 20. Placement of Decoupling Capacitors

ceramic, and polystyrene dielectrics work well to several hundred MHz. The other reason cited for not using too large a capacitance has to do with lead inductance.

The capacitor with its lead inductance forms a series LC circuit. Below the frequency of series resonance, the net impedance of the combination is capacitive. Above that frequency, the net impedance is inductive. Thus a decoupling capacitor is capacitive only below the frequency of series resonance. The frequency is given by

$$f_0 = \frac{1}{2\pi\sqrt{LC}}$$

where C is the decoupling capacitance and L is the lead inductance between the capacitor and the chip. On a PC board this inductance is determined by the layout, and is the same whether the capacitor dropped into the PCB holes is 0.001 μ F or 1 μ F. Thus, increasing the capacitance lowers the series resonant frequency. In fact, according to the resonant frequency formula, increasing C by a factor of 100 lowers the resonant frequency by a factor of 10.

Figures quoted on the series resonant frequency of a 0.01 μ F capacitor run from 10 to 15 MHz, depending on the lead length. If these numbers were accurate, a 1 μ F capacitor in the same position on the board would have a resonant frequency of 1.0 to 1.5 MHz, and as a decoupler would do more harm than good. However, the numbers are based on a presumed inductance of a given length of wire (the lead length). It should be noted that a "length of wire" has no inductance at all, strictly speaking. Only a complete current loop has inductance, and the inductance depends on the geometry of the loop. Figures quoted on the inductance of a length of wire are based on a presumably "very large" loop area, such that the magnetic field produced by the return current has no cancellation effect on the field produced by the current in the given length of wire. Such a loop geometry is not and should not be the case with the decoupling loop.

Figure 21 shows VCC waveforms, measured between pins 40 and 20 (VCC and VSS) of an 8751 CPU, for several conditions of decoupling on a PC board that has a decoupling loop area slightly larger than necessary. These photographs show the effects of increasing the decoupling capacitance and decreasing the area of the decoupling loop. The indications are that a 1 μ F capacitor is better than a 0.1 μ F capacitor, which in turn is better than nothing, and that the board should have been laid out with more attention paid to the area of the decoupling loop.

Figure 21e was obtained using a special-purpose experimental capacitor designed by Rogers Corp. (Q-Pac Division, Mesa, AZ) for use as a decoupler. It consists of two parallel plates, the length of a 40-pin DIP, separated by a ceramic dielectric. Sandwiched between the

CPU chip and the PCB (or between the CPU socket and the PCB), it makes connection to pins 40 and 20, forming a leadless decoupling capacitor. It is obviously a configuration of minimal inductance. Unfortunately, the particular sample tested had only 0.07 μ F of capacitance and so was unable to prevent the 1 MHz ripple as effectively as the configuration of Figure 21d. It seems apparent, though, that with more capacitance this part will alleviate a lot of decoupling problems.

THE CASE FOR ON-BOARD VOLTAGE REGULATION

To complicate matters, supply line glitches aren't always picked up in the distribution networks, but can come from the power supply circuit itself. In that case, a well-designed distribution network faithfully delivers the glitch throughout the system. The VCC glitch in Figure 22 was found to be coming from within a bench power supply in response to the EMP produced by an induction coil spark generator that was being used at Intel during a study of noise sensitivity. The VCC glitch is about 400 mV high and some 20 μ s in duration. Normal board decoupling techniques were ineffective in removing it, but adding an on-board voltage regulator chip did the job.

Thus, a good case can be made in favor of using a voltage regulator chip on each PCB, instead of doing all the voltage regulation at the supply circuit. This eases requirements on the heat-sinking at the supply circuit, and alleviates much of the distribution and board decoupling headaches. However, it also brings in the possibility that different boards would be operating at slightly different VCC levels due to tolerance in the regulator chips; this then leads to slightly different logic levels from board to board. The implications of that may vary from nothing to latch-up, depending on what kinds of chips are on the boards, and how they react to an input "high" that is perhaps 0.4V higher than local VCC.

Recovering Gracefully from a Software Upset

Even when one follows all the best guidelines for designing for a noisy environment, it's always possible for a noise transient to occur which exceeds the circuit's immunity level. In that case, one can strive at least for a graceful recovery.

Graceful recovery schemes involve additional hardware and/or software which is supposed to return the system to a normal operating mode after a software upset has occurred. Two decisions have to be made: How to recognize when an upset has occurred, and what to do about it.

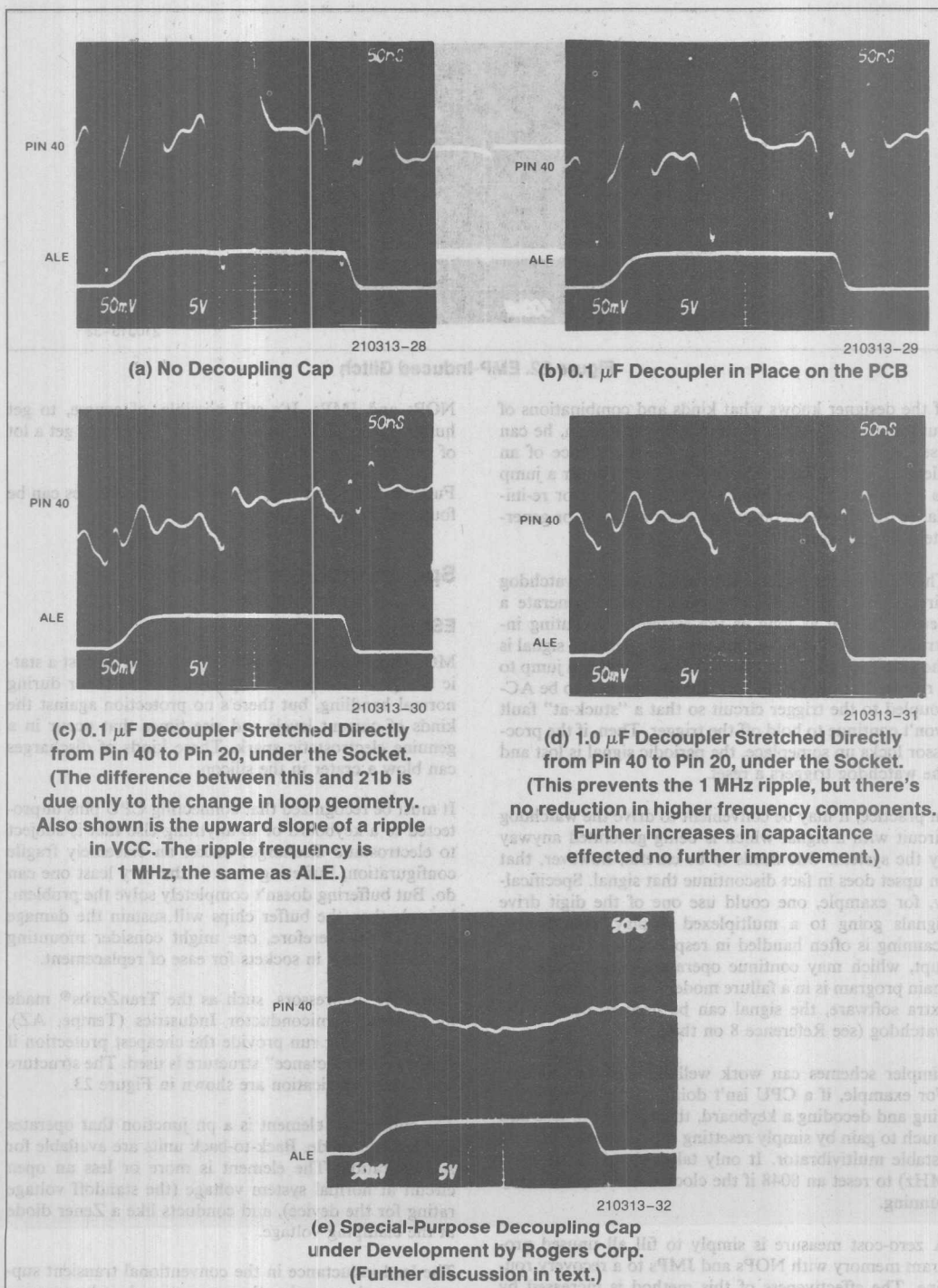


Figure 21. Noise on VCC Line

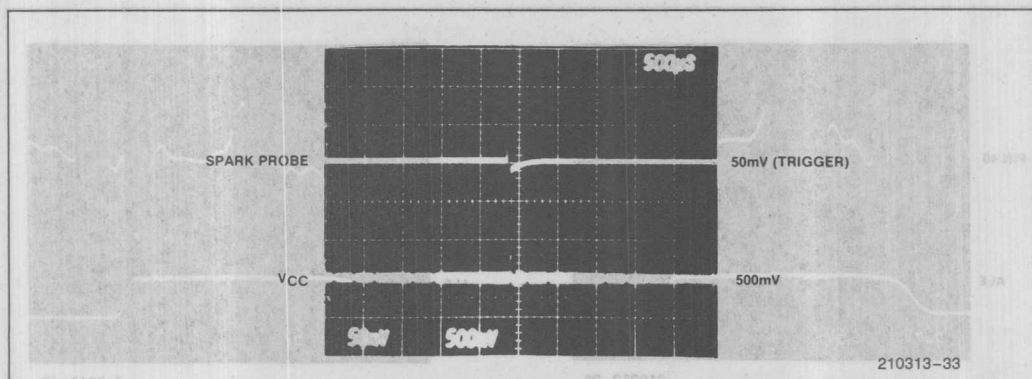


Figure 22. EMP-Induced Glitch

If the designer knows what kinds and combinations of outputs can legally be generated by the system, he can use gates to recognize and flag the occurrence of an illegal state of affairs. The flag can then trigger a jump to a recovery routine which then may check or re-initialize data, perhaps output an error message, or generate a simple reset.

The most reliable scheme is to use a so-called watchdog circuit. Here the CPU is programmed to generate a periodic signal as long as the system is executing instructions in an expected manner. The periodic signal is then used to hold off a circuit that will trigger a jump to a recovery routine. The periodic signal needs to be AC-coupled to the trigger circuit so that a "stuck-at" fault won't continue to hold off the trigger. Then, if the processor locks up someplace, the periodic signal is lost and the watchdog triggers a reset.

In practice, it may be convenient to drive the watchdog circuit with a signal which is being generated anyway by the system. One needs to be careful, however, that an upset does in fact discontinue that signal. Specifically, for example, one could use one of the digit drive signals going to a multiplexed display. But display scanning is often handled in response to a timer-interrupt, which may continue operating even though the main program is in a failure mode. Even so, with a little extra software, the signal can be used to control the watchdog (see Reference 8 on this).

Simpler schemes can work well for simpler systems. For example, if a CPU isn't doing anything but scanning and decoding a keyboard, there's little to lose and much to gain by simply resetting it periodically with an astable multivibrator. It only takes about 13 μ s (at 6 MHz) to reset an 8048 if the clock oscillator is already running.

A zero-cost measure is simply to fill all unused program memory with NOPs and JMPs to a recovery routine. The effectiveness of this method is increased by writing the program in segments that are separated by

NOPs and JMPs. It's still possible, of course, to get hung up in a data table or something. But you get a lot of protection, for the cost.

Further discussion of graceful recovery schemes can be found in Reference 13.

Special Problem Areas

ESD

MOS chips have some built-in protection against a static charge build-up on the pins, as would occur during normal handling, but there's no protection against the kinds of current levels and rise times that occur in a genuine electrostatic spark. These kinds of discharges can blow a crater in the silicon.

It must be recognized that connecting CPU pins unprotected to a keyboard or to anything else that is subject to electrostatic discharges makes an extremely fragile configuration. Buffering them is the very least one can do. But buffering doesn't completely solve the problem, because then the buffer chips will sustain the damage (even TTL); therefore, one might consider mounting the buffer chips in sockets for ease of replacement.

Transient suppressors, such as the TranZorbs® made by General Semiconductor Industries (Tempe, AZ), may in the long run provide the cheapest protection if their "zero inductance" structure is used. The structure and circuit application are shown in Figure 23.

The suppressor element is a pn junction that operates like a Zener diode. Back-to-back units are available for AC operation. The element is more or less an open circuit at normal system voltage (the standoff voltage rating for the device), and conducts like a Zener diode at the clamping voltage.

The lead inductance in the conventional transient suppressor package makes the conventional package essen-

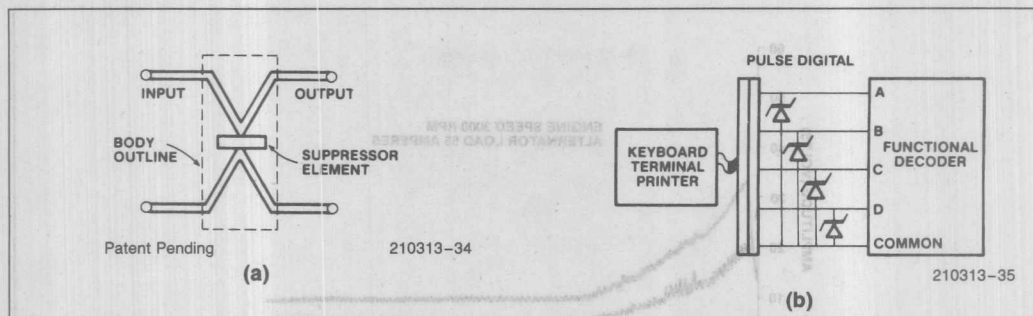


Figure 23. "Zero-Inductance" Structure and Use in Circuit

tially useless for protection against ESD pulses, owing to the fast rise of these pulses. The "zero inductance" units are available singly in a 4-pin DIP, and in arrays of four to a 16-pin DIP for PCB level protection. In that application they should be mounted in close proximity to the chips they protect.

In addition, metal enclosures or frames or parts that can receive an ESD spark should be connected by braided cable to the green-wire ground. Because of the ground impedance, ESD current shouldn't be allowed to flow through any signal ground, even if the chips are protected by transient suppressors. A 35 kV ESD spark can always spare a few hundred volts to drive a fast current pulse down a signal ground line if it can't find a braided cable to follow. Think how delighted your 8048 will be to find its VSS pin 250V higher than VCC for a few 10s of nanoseconds.

THE AUTOMOTIVE ENVIRONMENT

The automobile presents an extremely hostile environment for electronic systems. There are several parts to it:

1. Temperature extremes from -40°C to $+125^{\circ}\text{C}$ (under the hood) or $+85^{\circ}\text{C}$ (in the passenger compartment)
2. Electromagnetic pulses from the ignition system
3. Supply line transients that will knock your socks off

One needs to take a long, careful look at the temperature extremes. The allowable storage temperature range for most Intel MOS chips is -65°C to $+150^{\circ}\text{C}$, although some chips have a maximum storage temperature rating of $+125^{\circ}\text{C}$. In operation (or "under bias," as the data sheets say) the allowable ambient temperature range depends on the product grade, as follows:

Grade	Ambient Temperature	
	Min	Max
Commercial	0	70
Industrial	-40	$+85$
Automotive	-40	$+110$
Military	-55	$+125$

The different product grades are actually the same chip, but tested according to different standards. Thus, a given commercial-grade chip might actually pass military temperature requirements, but not have been tested for it. (Of course, there are other differences in grading requirements having to do with packaging, burn-in, traceability, etc.)

In any case, it's apparent that commercial-grade chips can't be used safely in automotive applications, not even in the passenger compartment. Industrial-grade chips can be used in the passenger compartment, and automotive or military chips are required in under-the-hood applications.

Ignition noise, CB radios, and that sort of thing are probably the least of your worries. In a poorly designed system, or in one that has not been adequately tested for the automotive environment, this type of EMI might cause a few software upsets, but not destroy chips.

The major problem, and the one that seems to come as the biggest surprise to most people, is the line transients. Regrettably, the 12V battery is not actually the source of power when the car is running. The charging system is, and it's not very clean. The only time the battery is the real source of power is when the car is first being started, and in that condition the battery terminals may be delivering about 5V or 6V. As follows is a brief description of the major idiosyncracies of the "12V" automotive power line.

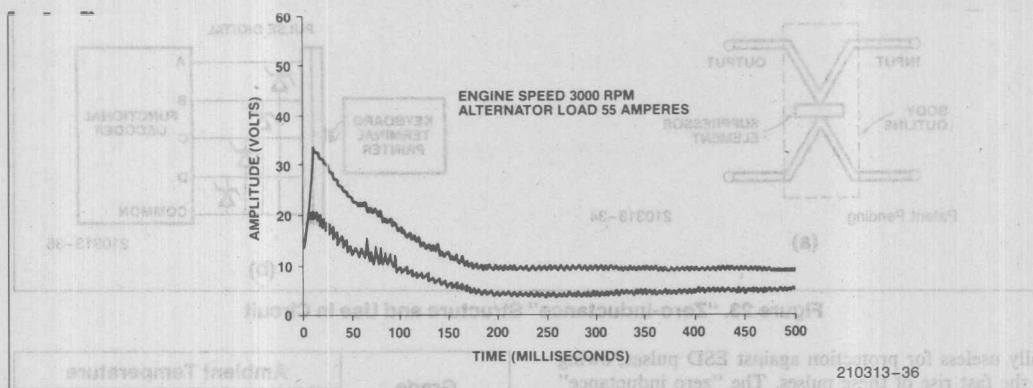


Figure 24. Typical Load Dump Transients

- An abrupt reduction in the alternator load causes a positive voltage transient called "load dump." In a load dump transient the line voltage rises to 20V or 30V in a few μ s, then decays exponentially with a time constant of about 100 μ s, as shown in Figure 24. Much higher peak voltages and longer decay times have also been reported. The worst case load dump is caused by disconnecting a low battery from the alternator circuit while the alternator is running. Normally this would happen intermittently when the battery terminal connections are defective.
- When the ignition is turned off, as the field excitation decays, the line voltage can go to between -40V and -100V for 100 μ s or more.
- Miscellaneous solenoid switching transients, such as the one shown in Figure 25, can drive the line to + or -200V to 400V for several μ s.

- Mutual coupling between unshielded wires in long harnesses can induce 100V and 200V transients in unprotected circuits.

What all this adds up to is that people in the business of building systems for automotive applications need a comprehensive testing program. An SAE guideline which describes the automotive environment is available to designers: SAE J1211, "Recommended Environmental Practices for Electronic Equipment Design," 1980 SAE Handbook, Part 1, pp. 22.80-22.96.

Some suggestions for protecting circuitry are shown in Figure 26. A transient suppressor is placed in front of the regulator chip to protect it. Since the rise times in these transients are not like those in ESD pulses, lead inductance is less critical and conventional devices can be used. The regulator itself is pretty much of a necessity, since a load dump transient is simply not going to be removed by any conventional LC or RC filter.

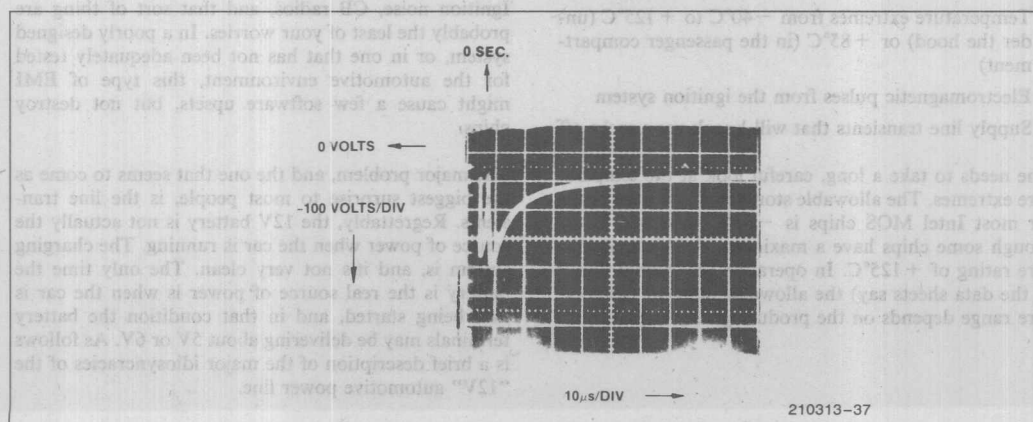


Figure 25. Transient Created by De-energizing an Air Conditioning Clutch Solenoid

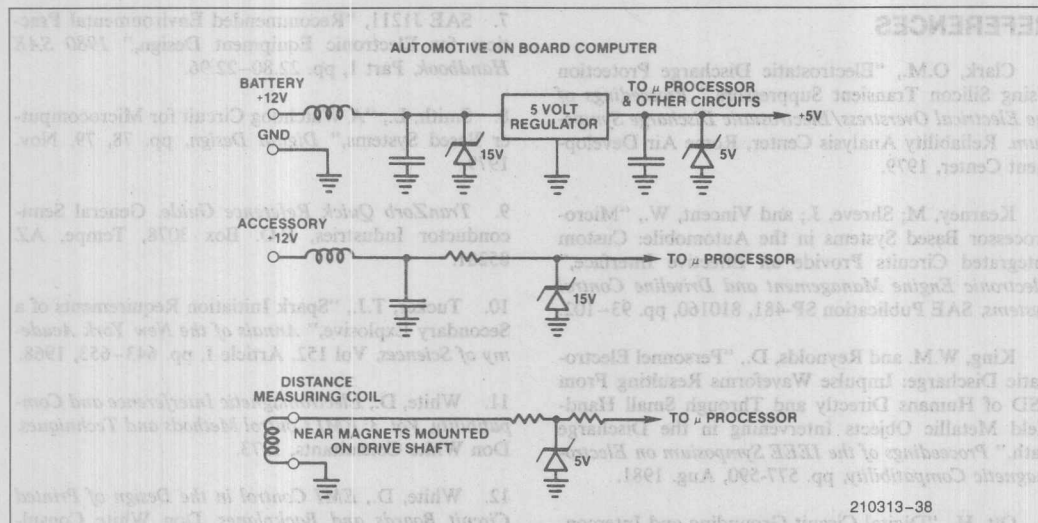


Figure 26. Use of Transient Suppressors in Automotive Applications

Special I/O interfacing is also required, because of the need for high tolerance to voltage transients, input noise, input/output isolation, etc. In addition, switches that are being monitored or driven by these buffers are usually referenced to chassis ground instead of signal ground, and in a car there can be many volts difference between the two. I/O interfacing is discussed in Reference 2.

Parting Thoughts

The main sources of information for this Application Note were the references by Ott and by White. Reference 5 is probably the finest treatment currently available on the subject. The other references provided specific information as cited in the text.

Courses and seminars on the subject of electromagnetic interference are given regularly throughout the year. Information on these can be obtained from:

IEEE Electromagnetic Compatibility Society
EMC Education Committee
345 East 47th Street
New York, NY 10017

Don White Consultants, Inc.
International Training Centre
P.O. Box D
Gainesville, VA 22065
Phone: (703) 347-0030

The EMC Education committee has available a video tape: "Introduction to EMC—A Video Training Tape," by Henry Ott. Don White Consultants offers a series of training courses on many different aspects of electromagnetic compatibility. Most organizations that sponsor EMC courses also offer in-plant presentations.

REFERENCES

1. Clark, O.M., "Electrostatic Discharge Protection Using Silicon Transient Suppressors," *Proceedings of the Electrical Overstress/Electrostatic Discharge Symposium*. Reliability Analysis Center, Rome Air Development Center, 1979.
2. Kearney, M; Shreve, J.; and Vincent, W., "Microprocessor Based Systems in the Automobile: Custom Integrated Circuits Provide an Effective Interface," *Electronic Engine Management and Driveline Control Systems*, SAE Publication SP-481, 810160, pp. 93-102.
3. King, W.M. and Reynolds, D., "Personnel Electrostatic Discharge: Impulse Waveforms Resulting From ESD of Humans Directly and Through Small Hand-Held Metallic Objects Intervening in the Discharge Path," *Proceedings of the IEEE Symposium on Electromagnetic Compatibility*, pp. 577-590, Aug. 1981.
4. Ott, H., "Digital Circuit Grounding and Interconnection," *Proceedings of the IEEE Symposium on Electromagnetic Compatibility*, pp. 292-297, Aug. 1981.
5. Ott, H., *Noise Reduction Techniques in Electronic Systems*. New York: Wiley, 1976.
6. 1981 *Interference Technology Engineers' Master (ITEM) Directory and Design Guide*. R. and B. Enterprises, P.O. Box 328, Plymouth Meeting, PA 19426.
7. SAE J1211, "Recommended Environmental Practices for Electronic Equipment Design," 1980 *SAE Handbook*, Part 1, pp. 22.80-22.96.
8. Smith, L., "A Watchdog Circuit for Microcomputer Based Systems," *Digital Design*, pp. 78, 79, Nov. 1979.
9. *TranZorb Quick Reference Guide*. General Semiconductor Industries, P.O. Box 3078, Tempe, AZ 85281.
10. Tucker, T.J., "Spark Initiation Requirements of a Secondary Explosive," *Annals of the New York Academy of Sciences*, Vol 152, Article I, pp. 643-653, 1968.
11. White, D., *Electromagnetic Interference and Compatibility, Vol. 3: EMI Control Methods and Techniques*. Don White Consultants, 1973.
12. White, D., *EMI Control in the Design of Printed Circuit Boards and Backplanes*. Don White Consultants, 1981.
13. Yarkoni, B. and Wharton, J., "Designing Reliable Software for Automotive Applications," *SAE Transactions*, 790237, July 1979.

an Note is intended to provide such as assistance in the design of oscillator circuits for microcontroller systems. Its purpose is to describe in a practical manner how oscillators work, how crystals and ceramic resonators work (and how to spec them), and what the on-chip amplifier looks like electrically. A program is provided in Appendix II to assist the designer in determining the effect of changing individual parameters. Suggestions are provided for optimizing a pre-production test program.

FEEDBACK OSCILLATORS

Loop Gain

Figure 1 shows an amplifier whose output line goes into some passive network. If the input signal to the amplifier is v_i , then the output signal from the amplifier is $v_o = A v_i$, and the output signal from the passive network is $v_f = B v_o = B A v_i$. Then $B A$ is the overall gain from terminal 1 to terminal 3.

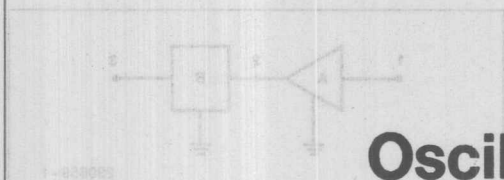


FIGURE 1 Loop Gain

Now connect terminal 1 to terminal 3 so that the signal path forms a loop. v_i to v_f which is also v_i . Now we have a feedback loop and the gain factor $B A$ is called the loop gain.

Gain factors are complex numbers. That means they have a magnitude and a phase angle both of which vary with frequency. When writing a complex number, one must specify both quantities, magnitude and angle. A number whose magnitude is 1, and whose angle is 0 degrees is commonly written this way: $1 \angle 0^\circ$. The number 1 is in complex number notation, $1 \angle 0^\circ$, while 1 is $1 \angle 0^\circ$.

By closing the feedback loop in Figure 1, we force the

$$v_f = 1 \cdot v_o$$

has two solutions

**TOM WILLIAMSON
MICROCONTROLLER
TECHNICAL MARKETING**

$$B A = 1 \angle 0^\circ$$

$$v_i = 0$$

Intel's microcontroller families (MCS-48, MCS-51, and iAPX-86) contain a circuit that is commonly referred to as the "on-chip oscillator." The on-chip circuit is not itself an oscillator, of course, but an amplifier that is suitable for use as the amplifier part of a feedback oscillator. The data sheets and Microcontroller Handbook show how the on-chip amplifier and several off-chip components can be used to design a working oscillator. With proper selection of off-chip components, these oscillator circuits will perform better than almost any other type of clock oscillator, and by almost any criterion of excellence. The suggested circuits are simple, economical, stable, and reliable.

December 1986

our customers in selecting suitable on-chip components to work with the on-chip oscillator circuit. It should be noted, however, that Intel cannot assume the responsibility of writing specifications for the off-chip components of the complete oscillator circuit, nor of guaranteeing the performance of the finished design in production, any more than a transistor manufacturer, whose data sheets show a number of suggested amplifier circuits, can assume responsibility for the operation in production of any of them.

We are often asked why we don't publish a list of recommended crystal or ceramic resonator specifications, and we are often asked why we don't publish a list of recommended values for the other off-chip components. This has been done in the past, but some of the consequences that were not intended.

Suppose we suggest a maximum value for some given frequency. This tells you the 30-ohm crystals are going to cost twice as much as 50-ohm crystals. Feeling that Intel will not "guarantee operation" with 50-ohm crystals, you order the expensive ones. In fact, Intel guarantees only what is embodied within its Intel product. Besides, there is no reason why 50-ohm crystals couldn't be used if the other off-chip components are suitably selected.

Should we recommend values for the other off-chip components? Should we do it for 30-ohm crystals or 50-ohm crystals? With respect to what should we optimize their selection? Should we minimize start-up time or maximize frequency stability? In many applications, neither start-up time nor frequency stability are particularly critical, and our "recommendations" are only intended to guide your system to unnecessary tolerances. It all depends on the application.

Although we will render "guidance," our "specific off-chip components" we do offer application engineers are intended to provide whatever technical assistance may be needed or desired by our customers in designing with Intel products.

INTRODUCTION

Intel's microcontroller families (MCS®-48, MCS®-51, and iACX-96) contain a circuit that is commonly referred to as the "on-chip oscillator". The on-chip circuitry is not itself an oscillator, of course, but an amplifier that is suitable for use as the amplifier part of a feedback oscillator. The data sheets and Microcontroller Handbook show how the on-chip amplifier and several off-chip components can be used to design a working oscillator. With proper selection of off-chip components, these oscillator circuits will perform better than almost any other type of clock oscillator, and by almost any criterion of excellence. The suggested circuits are simple, economical, stable, and reliable.

We offer assistance to our customers in selecting suitable off-chip components to work with the on-chip oscillator circuitry. It should be noted, however, that Intel cannot assume the responsibility of writing specifications for the off-chip components of the complete oscillator circuit, nor of guaranteeing the performance of the finished design in production, anymore than a transistor manufacturer, whose data sheets show a number of suggested amplifier circuits, can assume responsibility for the operation, in production, of any of them.

We are often asked why we don't publish a list of required crystal or ceramic resonator specifications, and recommend values for the other off-chip components. This has been done in the past, but sometimes with consequences that were not intended.

Suppose we suggest a maximum crystal resistance of 30 ohms for some given frequency. Then your crystal supplier tells you the 30-ohm crystals are going to cost twice as much as 50-ohm crystals. Fearing that Intel will not "guarantee operation" with 50-ohm crystals, you order the expensive ones. In fact, Intel guarantees only what is embodied within an Intel product. Besides, there is no reason why 50-ohm crystals couldn't be used, if the other off-chip components are suitably adjusted.

Should we recommend values for the other off-chip components? Should we do it for 50-ohm crystals or 30-ohm crystals? With respect to what should we optimize their selection? Should we minimize start-up time or maximize frequency stability? In many applications, neither start-up time nor frequency stability are particularly critical, and our "recommendations" are only restricting your system to unnecessary tolerances. It all depends on the application.

Although we will neither "specify" nor "recommend" specific off-chip components, we do offer assistance in these tasks. Intel application engineers are available to provide whatever technical assistance may be needed or desired by our customers in designing with Intel products.

This Application Note is intended to provide such assistance in the design of oscillator circuits for microcontroller systems. Its purpose is to describe in a practical manner how oscillators work, how crystals and ceramic resonators work (and thus how to spec them), and what the on-chip amplifier looks like electronically and what its operating characteristics are. A BASIC program is provided in Appendix II to assist the designer in determining the effects of changing individual parameters. Suggestions are provided for establishing a pre-production test program.

FEEDBACK OSCILLATORS

Loop Gain

Figure 1 shows an amplifier whose output line goes into some passive network. If the input signal to the amplifier is v_1 , then the output signal from the amplifier is $v_2 = Av_1$ and the output signal from the passive network is $v_3 = \beta v_2 = \beta Av_1$. Thus βA is the overall gain from terminal 1 to terminal 3.

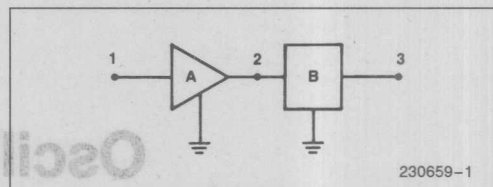


Figure 1. Factors in Loop Gain

Now connect terminal 1 to terminal 3, so that the signal path forms a loop: 1 to 2 to 3, which is also 1. Now we have a feedback loop, and the gain factor βA is called the *loop gain*.

Gain factors are complex numbers. That means they have a magnitude and a phase angle, both of which vary with frequency. When writing a complex number, one must specify both quantities, magnitude and angle. A number whose magnitude is 3, and whose angle is 45 degrees is commonly written this way: $3\angle 45^\circ$. The number 1 is, in complex number notation, $1\angle 0^\circ$, while -1 is $1\angle 180^\circ$.

By closing the feedback loop in Figure 1, we force the equality

$$v_1 = \beta Av_1$$

This equation has two solutions:

- 1) $v_1 = 0$;
- 2) $\beta A = 1\angle 0^\circ$.

In a given circuit, either or both of the solutions may be in effect. In the first solution the circuit is quiescent (no output signal). If you're trying to make an oscillator, a no-signal condition is unacceptable. There are ways to guarantee that the second solution is the one that will be in effect, and that the quiescent condition will be excluded.

How Feedback Oscillators Work

A feedback oscillator amplifies its own noise and feeds it back to itself in exactly the right phase, at the oscillation frequency, to build up and reinforce the desired oscillations. Its ability to do that depends on its loop gain. First, oscillations can occur only at the frequency for which the loop gain has a phase angle of 0 degrees. Second build-up of oscillations will occur only if the loop gain exceeds 1 at the frequency. Build-up continues until nonlinearities in the circuit reduce the average value of the loop gain to exactly 1.

Start-up characteristics depend on the small-signal properties of the circuit, specifically, the small-signal loop gain. Steady-state characteristics of the oscillator depend on the large-signal properties of the circuit, such as the transfer curve (output voltage vs. input voltage) of the amplifier, and the clamping effect of the input protection devices. These things will be discussed more fully further on. First we will look at the basic operation of the particular oscillator circuit, called the "positive reactance" oscillator.

The Positive Reactance Oscillator

Figure 2 shows the configuration of the positive reactance oscillator. The inverting amplifier, working into the impedance of the feedback network, produces an output signal that is nominally 180 degrees out of phase with its input. The feedback network must provide an additional 180 degrees phase shift, such that the overall loop gain has zero (or 360) degrees phase shift at the oscillation frequency.

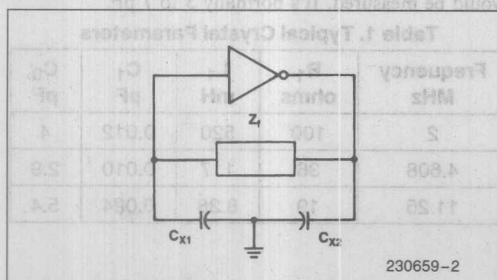


Figure 2. Positive Reactance Oscillator

In order for the loop gain to have zero phase angle it is necessary that the feedback element Z_f have a positive reactance. That is, it must be inductive. Then, the frequency at which the phase angle is zero is approximately the frequency at which

$$X_f = \frac{+1}{\omega C}$$

where X_f is the reactance of Z_f (the total Z_f being $R_f + jX_f$), and C is the series combination of C_{X1} and C_{X2} .

$$C = \frac{C_{X1} C_{X2}}{C_{X1} + C_{X2}}$$

In other words, Z_f and C form a parallel resonant circuit.

If Z_f is an inductor, then $X_f = \omega L$, and the frequency at which the loop gain has zero phase is the frequency at which

$$\omega L = \frac{1}{\omega C}$$

or

$$\omega = \frac{1}{\sqrt{LC}}$$

Normally, Z_f is not an inductor, but it must still have a positive reactance in order for the circuit to oscillate. There are some piezoelectric devices on the market that show a positive reactance, and provide a more stable oscillation frequency than an inductor will. Quartz crystals can be used where the oscillation frequency is critical, and lower cost ceramic resonators can be used where the frequency is less critical.

When the feedback element is a piezoelectric device, this circuit configuration is called a Pierce oscillator. The advantage of piezoelectric resonators lies in their property of providing a wide range of positive reactance values over a very narrow range of frequencies. The reactance will equal $1/\omega C$ at some frequency within this range, so the oscillation frequency will be within the same range. Typically, the width of this range is

only 0.3% of the nominal frequency of a quartz crystal, and about 3% of the nominal frequency of a ceramic resonator. With relatively little design effort, frequency accuracies of 0.03% or better can be obtained with quartz crystals, and 0.3% or better with ceramic resonators.

QUARTZ CRYSTALS

The crystal resonator is a thin slice of quartz sandwiched between two electrodes. Electrically, the device looks pretty much like a 5 or 6 pF capacitor, except that over certain ranges of frequencies the crystal has a positive (i.e., inductive) reactance.

The ranges of positive reactance originate in the piezoelectric property of quartz: Squeezing the crystal generates an internal E-field. The effect is reversible: Applying an AC E-field causes the crystal to vibrate. At certain vibrational frequencies there is a mechanical resonance. As the E-field frequency approaches a frequency of mechanical resonance, the measured reactance of the crystal becomes positive, as shown in Figure 3.

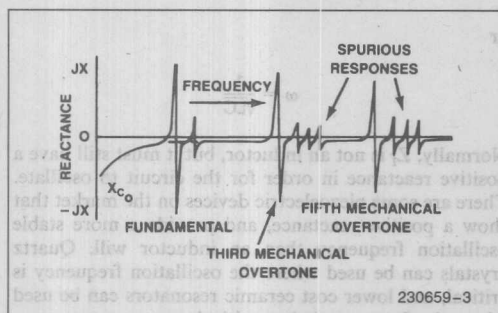


Figure 3. Crystal Reactance vs. Frequency

Typically there are several ranges of frequencies where in the reactance of the crystal is positive. Each range corresponds to a different mode of vibration in the crystal. The main resonances are the so-called fundamental response and the third and fifth overtone responses.

The overtone responses shouldn't be confused with the harmonics of the fundamental. They're not harmonics, but different vibrational modes. They're not in general at exact integer multiples of the fundamental frequency. There will also be "spurious" responses, occurring typically a few hundred KHz above each main response.

To assure that an oscillator starts in the desired mode on power-up, something must be done to suppress the loop gain in the undesired frequency ranges. The crystal itself provides some protection against unwanted modes of oscillation; too much resistance in that mode, for example. Additionally, junction capacitances in the amplifying devices tend to reduce the gain at higher frequencies, and thus may discriminate against unwanted modes. In some cases a circuit fix is necessary, such as inserting a trap, a phase shifter, or ferrite beads to kill oscillations in unwanted modes.

Crystal Parameters

Equivalent Circuit

Figure 4 shows an equivalent circuit that is used to represent the crystal for circuit analysis.

The R_1 - L_1 - C_1 branch is called the motivational arm of the crystal. The values of these parameters derive from the mechanical properties of the crystal and are constant for a given mode of vibration. Typical values for various nominal frequencies are shown in Table 1.

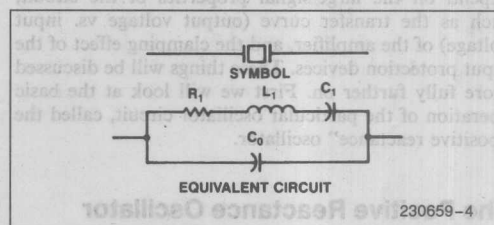


Figure 4. Quartz Crystal: Symbol and Equivalent Circuit

C_0 is called the shunt capacitance of the crystal. This is the capacitance of the crystal's electrodes and the mechanical holder. If one were to measure the reactance of the crystal at a frequency far removed from a resonance frequency, it is the reactance of this capacitance that would be measured. It's normally 3 to 7 pF.

Table 1. Typical Crystal Parameters

Frequency MHz	R_1 ohms	L_1 mH	C_1 pF	C_0 pF
2	100	520	0.012	4
4.608	36	117	0.010	2.9
11.25	19	8.38	0.024	5.4

The series resonant frequency of the crystal is the frequency at which L_1 and C_1 are in resonance. This frequency is given by

$$f_s = \frac{1}{2\pi\sqrt{L_1 C_1}}$$

At this frequency the impedance of the crystal is R_1 in parallel with the reactance of C_0 . For most purposes, this impedance is taken to be just R_1 , since the reactance of C_0 is so much larger than R_1 .

Load Capacitance

A crystal oscillator circuit such as the one shown in Figure 2 (redrawn in Figure 5) operates at the frequency for which the crystal is antiresonant (ie, parallel-resonant) with the total capacitance across the crystal terminals external to the crystal. This total capacitance external to the crystal is called the load capacitance.

As shown in Figure 5, the load capacitance is given by

$$C_L = \frac{C_{X1} C_{X2}}{C_{X1} + C_{X2}} + C_{\text{stray}}$$

The crystal manufacturer needs to know the value of C_L in order to adjust the crystal to the specified frequency.

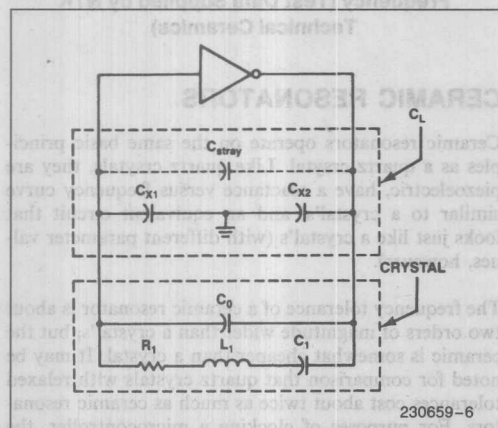


Figure 5. Load Capacitance

The adjustment involves putting the crystal in series with the specified C_L , and then "trimming" the crystal to obtain resonance of the series combination of the crystal and C_L at the specified frequency. Because of the high Q of the crystal, the resonant frequency of the series combination of the crystal and C_L is the same as

the antiresonant frequency of the parallel combination of the crystal and C_L . This frequency is given by

$$f_a = \frac{1}{2\pi\sqrt{L_1 C_1 (C_L + C_0) / (C_1 + C_L + C_0)}}$$

These frequency formulas are derived (in Appendix A) from the equivalent circuit of the crystal, using the assumptions that the Q of the crystal is extremely high, and that the circuit external to the crystal has no effect on the frequency other than to provide the load capacitance C_L . The latter assumption is not precisely true, but it is close enough for present purposes.

"Series" vs. "Parallel" Crystals

There is no such thing as a "series cut" crystal as opposed to a "parallel cut" crystal. There are different cuts of crystal, having to do with the parameters of its motional arm in various frequency ranges, but there is no special cut for series or parallel operation.

An oscillator is series resonant if the oscillation frequency is f_s of the crystal. To operate the crystal at f_s , the amplifier has to be noninverting. When buying a crystal for such an oscillator, one does not specify a load capacitance. Rather, one specifies the loading condition as "series."

If a "series" crystal is put into an oscillator that has an inverting amplifier, it will oscillate in parallel resonance with the load capacitance presented to the crystal by the oscillator circuit, at a frequency slightly above f_s . In fact, at approximately

$$f_a = f_s \left(1 + \frac{C_1}{2(C_L + C_0)} \right)$$

This frequency would typically be about 0.02% above f_s .

Equivalent Series Resistance

The "series resistance" often listed on quartz crystal data sheets is the real part of the crystal impedance at the crystal's calibration frequency. This will be R_1 if the calibration frequency is the series resonant frequency of the crystal. If the crystal is calibrated for parallel resonance with a load capacitance C_L , the equivalent series resistance will be

$$ESR = R_1 \left(1 + \frac{C_0}{C_L} \right)^2$$

The crystal manufacturer measures this resistance at the calibration frequency during the same operation in which the crystal is adjusted to the calibration frequency.

Frequency Tolerance

Frequency tolerance as discussed here is not a requirement on the crystal, but on the complete oscillator. There are two types of frequency tolerances on oscillators: frequency *accuracy* and frequency *stability*. Frequency accuracy refers to the oscillator's ability to run at an exact specified frequency. Frequency stability refers to the constancy of the oscillation frequency.

Frequency accuracy requires mainly that the oscillator circuit present to the crystal the same load capacitance that it was adjusted for. Frequency stability requires mainly that the load capacitance be constant.

In most digital applications the accuracy and stability requirements on the oscillator are so wide that it makes very little difference what load capacitance the crystal was adjusted to, or what load capacitance the circuit actually presents to the crystal. For example, if a crystal was calibrated to a load capacitance of 25 pF, and is used in a circuit whose actual load capacitance is 50 pF, the frequency error on that account would be less than 0.01%.

In a positive reactance oscillator, the crystal only needs to be in the intended response mode for the oscillator to satisfy a 0.5% or better frequency tolerance. That's because for any load capacitance the oscillation frequency is certain to be between the crystal's resonant and anti-resonant frequencies.

Phase shifts that take place within the amplifier part of the oscillator will also affect frequency accuracy and stability. These phase shifts can normally be modeled as an "output capacitance" that, in the positive reactance oscillator, parallels C_{X2} . The predictability and constancy of this output capacitance over temperature and device sample will be the limiting factor in determining the tolerances that the circuit is capable of holding.

Drive Level

Drive level refers to the power dissipation in the crystal. There are two reasons for specifying it. One is that the parameters in the equivalent circuit are somewhat dependent on the drive level at which the crystal is calibrated. The other is that if the application circuit exceeds the test drive level by too much, the crystal may be damaged. Note that the terms "test drive level" and "rated drive level" both refer to the drive level at which the crystal is calibrated. Normally, in a microcontroller system, neither the frequency tolerances nor the power levels justify much concern for this specification. Some crystal manufacturers don't even require it for microprocessor crystals.

In a positive reactance oscillator, if one assumes the peak voltage across the crystal to be something in the neighborhood of V_{CC} , the power dissipation can be approximated as

$$P = 2R_1 [\pi f (C_L + C_0) V_{CC}]^2$$

This formula is derived in Appendix A. In a 5V system, P rarely evaluates to more than a milliwatt. Crystals with a standard 1 or 2 mW drive level rating can be used in most digital systems.

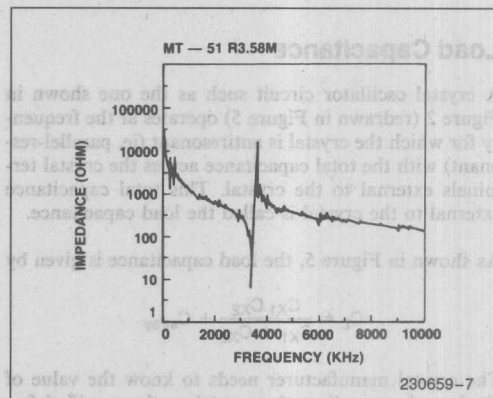


Figure 6. Ceramic Resonator Impedance vs. Frequency (Test Data Supplied by NTK Technical Ceramics)

CERAMIC RESONATORS

Ceramic resonators operate on the same basic principles as a quartz crystal. Like quartz crystals, they are piezoelectric, have a reactance versus frequency curve similar to a crystal's, and an equivalent circuit that looks just like a crystal's (with different parameter values, however).

The frequency tolerance of a ceramic resonator is about two orders of magnitude wider than a crystal's, but the ceramic is somewhat cheaper than a crystal. It may be noted for comparison that quartz crystals with relaxed tolerances cost about twice as much as ceramic resonators. For purposes of clocking a microcontroller, the frequency tolerance is often relatively noncritical, and the economic consideration becomes the dominant factor.

Figure 6 shows a graph of impedance magnitude versus frequency for a 3.58 MHz ceramic resonator. (Note that Figure 6 is a graph of $|Z_f|$ versus frequency, where

as Figure 3 is a graph of X_f versus frequency.) A number of spurious responses are apparent in Figure 6. The manufacturers state that spurious responses are more prevalent in the lower frequency resonators (kHz range) than in the higher frequency units (MHz range). For our purposes only the MHz range ceramics need to be considered.

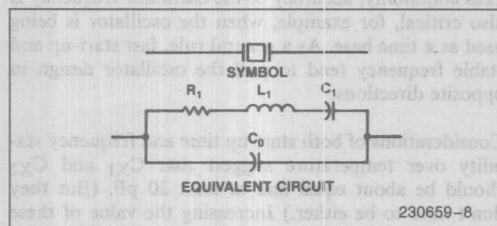


Figure 7. Ceramic Resonator: Symbol and Equivalent Circuit

Figure 7 shows the symbol and equivalent circuit for the ceramic resonator, both of which are the same as for the crystal. The parameters have different values, however, as listed in Table 2.

Table 2. Typical Ceramic Parameters

Frequency MHz	R_1 ohms	L_1 mH	C_1 pF	C_0 pF
3.58	7	0.113	19.6	140
6.0	8	0.094	8.3	60
8.0	7	0.092	4.6	40
11.0	10	0.057	3.9	30

Note that the motional arm of the ceramic resonator tends to have less resistance than the quartz crystal and also a vastly reduced L_1/C_1 ratio. This results in the motional arm having a Q (given by $(1/R_1) \sqrt{L_1/C_1}$) that is typically two orders of magnitude lower than that of a quartz crystal. The lower Q makes for a faster startup of the oscillator and for a less closely controlled frequency (meaning that circuitry external to the resonator will have more influence on the frequency than with a quartz crystal).

Another major difference is that the shunt capacitance of the ceramic resonator is an order of magnitude higher than C_0 of the quartz crystal and more dependent on the frequency of the resonator.

The implications of these differences are not all obvious, but some will be indicated in the section on Oscillator Calculations.

Specifications for Ceramic Resonators

Ceramic resonators are easier to specify than quartz crystals. All the vendor wants to know is the desired

frequency and the chip you want it to work with. They'll supply the resonators, a circuit diagram showing the positions and values of other external components that may be required and a guarantee that the circuit will work properly at the specified frequency.

OSCILLATOR DESIGN CONSIDERATIONS

Designers of microcontroller systems have a number of options to choose from for clocking the system. The main decision is whether to use the "on-chip" oscillator or an external oscillator. If the choice is to use the on-chip oscillator, what kinds of external components are needed to make it operate as advertised? If the choice is to use an external oscillator, what type of oscillator should it be?

The decisions have to be based on both economic and technical requirements. In this section we'll discuss some of the factors that should be considered.

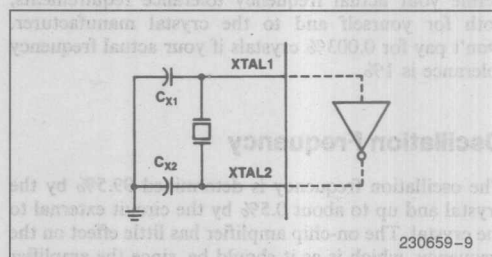


Figure 8. Using the "On-Chip" Oscillator

On-Chip Oscillators

In most cases, the on-chip amplifier with the appropriate external components provides the most economical solution to the clocking problem. Exceptions may arise in severe environments when frequency tolerances are tighter than about 0.01%.

The external components that need to be added are a positive reactance (normally a crystal or ceramic resonator) and the two capacitors C_{X1} and C_{X2} , as shown in Figure 8.

Crystal Specifications

Specifications for an appropriate crystal are not very critical, unless the frequency is. Any fundamental-mode crystal of medium or better quality can be used.

should be specified. The best answer to this question is the lower the better, but use what's available. The crystal resistance will have some effect on start-up time and steady-state amplitude, but not so much that it can't be compensated for by appropriate selection of the capacitances C_{X1} and C_{X2} .

Similar questions are asked about specifications of load capacitance and shunt capacitance. The best advice we can give is to understand what these parameters mean and how they affect the operation of the circuit (that being the purpose of this Application Note), and then decide for yourself if such specifications are meaningful in your application or not. Normally, they're not, unless your frequency tolerances are tighter than about 0.1%.

Part of the problem is that crystal manufacturers are accustomed to talking "ppm" tolerances with radio engineers and simply won't take your order until you've filled out their list of specifications. It will help if you define your actual frequency tolerance requirements, both for yourself and to the crystal manufacturer. Don't pay for 0.003% crystals if your actual frequency tolerance is 1%.

Oscillation Frequency

The oscillation frequency is determined 99.5% by the crystal and up to about 0.5% by the circuit external to the crystal. The on-chip amplifier has little effect on the frequency, which is as it should be, since the amplifier parameters are temperature and process dependent.

The influence of the on-chip amplifier on the frequency is by means of its input and output (pin-to-ground) capacitances, which parallel C_{X1} and C_{X2} , and the XTAL1-to-XTAL2 (pin-to-pin) capacitance, which parallels the crystal. The input and pin-to-pin capacitances are about 7 pF each. Internal phase deviations from the nominal 180° can be modeled as an output capacitance of 25 to 30 pF. These deviations from the ideal have less effect in the positive reactance oscillator (with the inverting amplifier) than in a comparable series resonant oscillator (with the noninverting amplifier) for two reasons: first, the effect of the output capacitance is lessened, if not swamped, by the off-chip capacitor; secondly, the positive reactance oscillator is less sensitive, frequency-wise, to such phase errors.

Selection of C_{X1} and C_{X2}

Optimal values for the capacitors C_{X1} and C_{X2} depend on whether a quartz crystal or ceramic resonator

requirements on start-up time and frequency tolerance. Start-up time is sometimes more critical in microcontroller systems than frequency stability, because of various reset and initialization requirements.

Less commonly, accuracy of the oscillator frequency is also critical, for example, when the oscillator is being used as a time base. As a general rule, fast start-up and stable frequency tend to pull the oscillator design in opposite directions.

Considerations of both start-up time and frequency stability over temperature suggest that C_{X1} and C_{X2} should be about equal and at least 20 pF. (But they don't have to be either.) Increasing the value of these capacitances above some 40 or 50 pF improves frequency stability. It also tends to increase the start-up time. There is a maximum value (several hundred pF, depending on the value of R_1 of the quartz or ceramic resonator) above which the oscillator won't start up at all.

If the on-chip amplifier is a simple inverter, such as in the 8051, the user can select values for C_{X1} and C_{X2} between some 20 and 100 pF, depending on whether start-up time or frequency stability is the more critical parameter in a specific application. If the on-chip amplifier is a Schmitt Trigger, such as in the 8048, smaller values of C_{X1} must be used (5 to 30 pF), in order to prevent the oscillator from running in a relaxation mode.

Later sections in this Application Note will discuss the effects of varying C_{X1} and C_{X2} (as well as other parameters), and will have more to say on their selection.

Placement of Components

Noise glitches arriving at XTAL1 or XTAL2 pins at the wrong time can cause a miscount in the internal clock-generating circuitry. These kinds of glitches can be produced through capacitive coupling between the oscillator components and PCB traces carrying digital signals with fast rise and fall times. For this reason, the oscillator components should be mounted close to the chip and have short, direct traces to the XTAL1, XTAL2, and VSS pins.

Clocking Other Chips

There are times when it would be desirable to use the on-chip oscillator to clock other chips in the system.

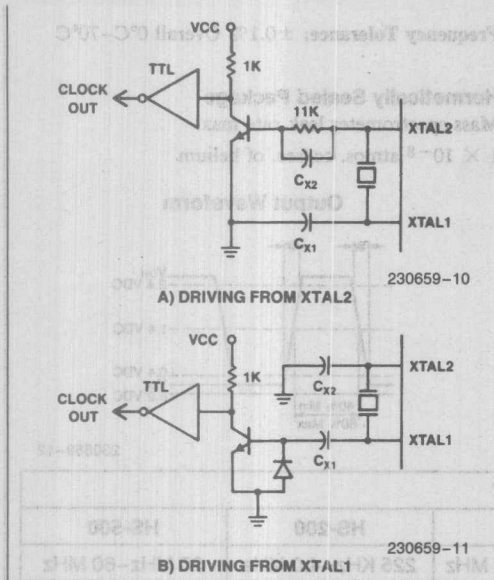


Figure 9. Using the On-Chip Oscillator to Drive Other Chips

OUTPUT				
HS-100	HS-100	HS-100	HS-100	
2.5 MHz-30 MHz	2.5 MHz-30 MHz	2.5 MHz-30 MHz	2.5 MHz-30 MHz	V _{OH} (Logic "1")
+2.5V min	+2.5V min	+2.5V min	+2.5V min	V _{OL} (Logic "0")
+0.4V max	+0.4V max	+0.4V max	+0.4V max	Symmetry
60% min	60% min	60% min	60% min	T _r , T _f (Free A)
< 2 ns	< 2 ns	< 2 ns	< 2 ns	Fall Time
1 to 10 T _T Load	1 to 10 T _T Load	1 to 10 T _T Load	1 to 10 T _T Load	Output Short
18 mA min	18 mA min	18 mA min	18 mA min	Output Current
1 to 10 T _T Load	1 to 10 T _T Load	1 to 10 T _T Load	1 to 10 T _T Load	Output Load
CONDITIONS				
V _{DD} = 5.0V, V _{SS} = 0V, T _A = 25°C				
I _{DD} sink = 18.0 mA max				
I _{DD} source = 1.0 mA max				
V _{DD} = 5.0V, V _{SS} = 0V, T _A = 25°C				
I _{DD} sink = 30.0 mA max				
I _{DD} source = 1.8 mA max				

Figure 10. Pre-Packaged Oscillator Data

TTL buffer puts too much load on the on-chip amplifier for reliable start-up. A CMOS buffer (such as the 74HC04) can be used, if it's fast enough and if its V_{IH} and V_{IL} specs are compatible with the available signal amplitudes. Circuits such as shown in Figure 9 might also be considered for these types of applications.

Clock-related signals are available at the TO pin in the MCS-48 products, at ALE in the MCS-48 and MCS-51 lines, and the iACX-96 controllers provide a CLKOUT signal.

External Oscillators

When technical requirements dictate the use of an external oscillator, the external drive requirements for the microcontroller, as published in the data sheet, must be carefully noted. The logic levels are not in general TTL-compatible. And each controller has its idiosyncracies in this regard. The 8048, for example, requires that both XTAL1 and XTAL2 be driven. The 8051 can be driven that way, but the data sheet suggest the simpler method of grounding XTAL1 and driving XTAL2. For this method, the driving source must be capable of sinking some current when XTAL2 is being driven low.

For the external oscillator itself, there are basically two choices: ready-made and home-grown.

TTL Crystal Clock Oscillator

The HS-100, HS-200, & HS-500 all-metal package series of oscillators are TTL compatible & fit a DIP layout. Standard electrical specifications are shown below. Variations are available for special applications.

Frequency Range: HS-100—3.5 MHz to 30 MHz

HS-200—225 KHz to 3.5 MHz

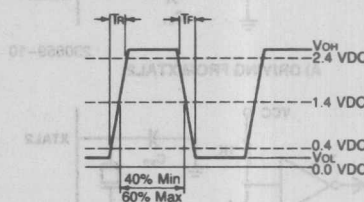
HS-500—25 MHz to 60 MHz

Frequency Tolerance: $\pm 0.1\%$ Overall $0^{\circ}\text{C}-70^{\circ}\text{C}$

Hermetically Sealed Package

Mass spectrometer leak rate max.

1×10^{-8} atmos. cc/sec. of helium

Output Waveform

230659-12

INPUT				
	HS-100		HS-200	HS-500
	3.5 MHz-20 MHz	20 + MHz-30 MHz	225 KHz-4.0 MHz	25 MHz-60 MHz
Supply Voltage (V_{CC})	$5V \pm 10\%$	$5V \pm 10\%$	$5V \pm 10\%$	$5V \pm 10\%$
Supply Current (I_{CC}) max.	30 mA	40 mA	85 mA	50 mA
OUTPUT				
	HS-100		HS-200	HS-500
	3.5 MHz-20 MHz	20 + MHz-30 MHz	225 KHz-4.0 MHz	25 MHz-60 MHz
V_{OH} (Logic "1")	+2.4V min. ¹	+2.7V min. ²	+2.4V min. ¹	+2.7V min. ²
V_{OL} (Logic "0")	+0.4V max. ³	+0.5V max. ⁴	+0.4V max. ³	+0.5V max. ⁴
Symmetry	60/40% ⁵	60/40% ⁵	55/45% ⁵	60/40% ⁵
T_R , T_F (Rise & Fall Time)	< 10 ns ⁶	< 5 ns ⁶	< 15 ns ⁶	< 5 ns ⁶
Output Short Circuit Current	18 mA min.	40 mA min.	18 mA min.	40 mA min.
Output Load	1 to 10 TTL Loads ⁷	1 to 10 TTL Loads ⁸	1 to 10 TTL Loads ⁷	1 to 10 TTL Loads ⁸
CONDITIONS				
¹ I_O source = -400 μA max.	⁴ I_O sink = 20.00 mA max.		71.6 mA per load	
² I_O source = -1.0 mA max.	⁵ $V_O = 1.4V$		⁸ 2.0 mA per load	
³ I_O sink = 16.0 mA max.	⁶ (0.4V to 2.4V)			

Figure 10. Pre-Packaged Oscillator Data*

Prepackaged oscillators are available from most crystal manufacturers, and have the advantage that the system designer can treat the oscillator as a black box whose performance is guaranteed by people who carry many years of experience in designing and building oscillators. Figure 10 shows a typical data sheet for some prepackaged oscillators. Oscillators are also available with complementary outputs.

If the oscillator is to drive the microcontroller directly, one will want to make a careful comparison between the external drive requirements in the microcontroller data sheet and the oscillator's output logic levels and test conditions.

If oscillator stability is less critical than cost, the user may prefer to go with an in-house design. Not without some precautions, however.

It's easy to design oscillators that work. Almost all of them do work, even if the designer isn't too clear on why. The key point here is that *almost* all of them work. The problems begin when the system goes into production, and marginal units commence malfunctioning in the field. Most digital designers, after all, are not very adept at designing oscillators for production.

Oscillator design is somewhat of a black art, with the quality of the finished product being *very* dependent on the designer's experience and intuition. For that reason the most important consideration in any design is to have an adequate preproduction test program. Preproduction tests are discussed later in this Application Note. Here we will discuss some of the design options and take a look at some commonly used configurations.

Gate Oscillators versus Discrete Devices

Digital systems designers are understandably reluctant to get involved with discrete devices and their peculiarities (biasing techniques, etc.). Besides, the component count for these circuits tends to be quite a bit higher than what a digital designer is used to seeing for that amount of functionality. Nevertheless, if there are unusual requirements on the accuracy and stability of the clock frequency, it should be noted that discrete device oscillators can be tailored to suit the exact needs of the application and perfected to a level that would be difficult for a gate oscillator to approach.

In most cases, when an external oscillator is needed, the designer tends to rely on some form of a gate oscillator. A TTL inverter with a resistor connecting the output to the input makes a suitable inverting amplifier. The resistor holds the inverter in the transition region between logical high and low, so that at least for start-up purposes the inverter is a linear amplifier.

The feedback resistance has to be quite low, however, since it must conduct current sourced by the input pin without allowing the DC input voltage to get too far above the DC output voltage. For biasing purposes, the feedback resistance should not exceed a few k-ohms. But shunting the crystal with such a low resistance does not encourage start-up.

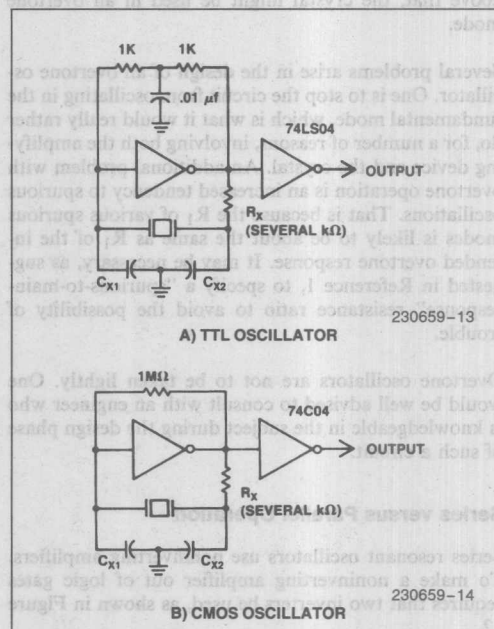


Figure 11. Commonly Used Gate Oscillators

Consequently, the configuration in Figure 11A might be suggested. By breaking R_F into two parts and AC-grounding the midpoint, one achieves the DC feedback required to hold the inverter in its active region, but without the negative signal feedback that is in effect telling the circuit *not* to oscillate. However, this biasing scheme will increase the start-up time, and relaxation-type oscillations are also possible.

A CMOS inverter, such as the 74HC04, might work better in this application, since a larger R_F can be used to hold the inverter in its linear region.

Logic gates tend to have a fairly low output resistance, which destabilizes the oscillator. For that reason a resistor R_X is often added to the feedback network, as shown in Figures 11A and B. At higher frequencies a 20 or 30 pF capacitor is sometimes used in the R_X position, to compensate for some of the internal propagation delay.

Reference 1 contains an excellent discussion of gate oscillators, and a number of design examples.

Fundamental versus Overtone Operation

It's easier to design an oscillator circuit to operate in the resonator's fundamental response mode than to design one for overtone operation. A quartz crystal whose fundamental response mode covers the desired frequency can be obtained up to some 30 MHz. For frequencies above that, the crystal might be used in an overtone mode.

Several problems arise in the design of an overtone oscillator. One is to stop the circuit from oscillating in the fundamental mode, which is what it would really rather do, for a number of reasons, involving both the amplifying device and the crystal. An additional problem with overtone operation is an increased tendency to spurious oscillations. That is because the R_1 of various spurious modes is likely to be about the same as R_1 of the intended overtone response. It may be necessary, as suggested in Reference 1, to specify a "spurious-to-main-response" resistance ratio to avoid the possibility of trouble.

Overtone oscillators are not to be taken lightly. One would be well advised to consult with an engineer who is knowledgeable in the subject during the design phase of such a circuit.

Series versus Parallel Operation

Series resonant oscillators use noninverting amplifiers. To make a noninverting amplifier out of logic gates requires that two inverters be used, as shown in Figure 12.

This type of circuit tends to be inaccurate and unstable in frequency over variations in temperature and V_{CC} . It has a tendency to oscillate at overtones, and to oscillate through C_0 of the crystal or some stray capacitance rather than as controlled by the mechanical resonance of the crystal.

The demon in series resonant oscillators is the phase shift in the amplifier. The series resonant oscillator wants more than just a "noninverting" amplifier—it wants a *zero phase-shift* amplifier. Multistage noninverting amplifiers tend to have a considerably lagging phase shift, such that the crystal reactance must be capacitive in order to bring the total phase shift around the feedback loop back up to 0. In this mode, a "12 MHz" crystal may be running at 8 or 9 MHz. One can put a capacitor in series with the crystal to relieve the crystal of having to produce all of the required phase shift, and bring the oscillation frequency closer to fs. However, to further complicate the situation, the amplifier's phase shift is strongly dependent on frequency, temperature, VCC, and device sample.

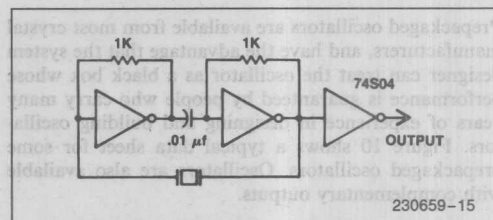


Figure 12. "Series Resonant" Gate Oscillator

Positive reactance oscillators ("parallel resonant") use inverting amplifiers. A single logic inverter can be used for the amplifier, as in Figure 11. The amplifier's phase shift is less critical, compared to a series resonant circuit, and since only one inverter is involved there's less phase error anyway. The oscillation frequency is effectively bounded by the resonant and antiresonant frequencies of the crystal itself. In addition, the feedback network includes capacitors that parallel the input and output terminals of the amplifier, thus reducing the effect of unpredictable capacitances at these points.

MORE ABOUT USING THE "ON-CHIP" OSCILLATORS

In this section we will describe the on-chip inverters on selected microcontrollers in some detail, and discuss criteria for selecting components to work with them. Future data sheets will supplement this discussion with updates and information pertinent to the use of each chip's oscillator circuitry.

Oscillator Calculations

Oscillator design, though aided by theory, is still largely an empirical exercise. The circuit is inherently nonlinear, and the normal analysis parameters vary with instantaneous voltage. In addition, when dealing with the on-chip circuitry, we have FETs being used as resistors, resistors being used as interconnects, distributed delays, input protection devices, parasitic junctions, and processing variations.

Consequently, oscillator calculations are never very precise. They can be useful, however, if they will at least indicate the effects of variations in the circuit parameters on start-up time, oscillation frequency, and steady-state amplitude. Start-up time, for example, can be taken as an indication of start-up reliability. If pre-production tests indicate a possible start-up problem, a relatively inexperienced designer can at least be made aware of what parameter may be causing the marginality, and what direction to go in to fix it.

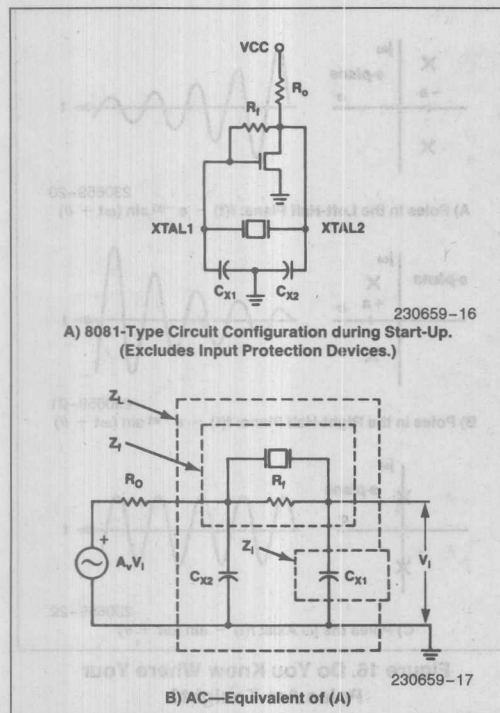


Figure 13. Oscillator Circuit Model Used in Start-Up Calculations

The analysis used here is mathematically straightforward but algebraically intractable. That means it's relatively easy to understand and program into a computer, but it will not yield a neat formula that gives, say, steady-state amplitude as a function of this or that list of parameters. A listing of a BASIC program that implements the analysis will be found in Appendix II.

When the circuit is first powered up, and before the oscillations have commenced (and if the oscillations fail to commence), the oscillator can be treated as a small signal linear amplifier with feedback. In that case, standard small-signal analysis techniques can be used to determine start-up characteristics. The circuit model used in this analysis is shown in Figure 13.

The circuit approximates that there are no high-frequency effects within the amplifier itself, such that its high-frequency behavior is dominated by the load impedance Z_L . This is a reasonable approximation for single-stage amplifiers of the type used in 8051-type devices. Then the gain of the amplifier as a function of frequency is

$$A = \frac{A_v Z_L}{Z_L + R_o}$$

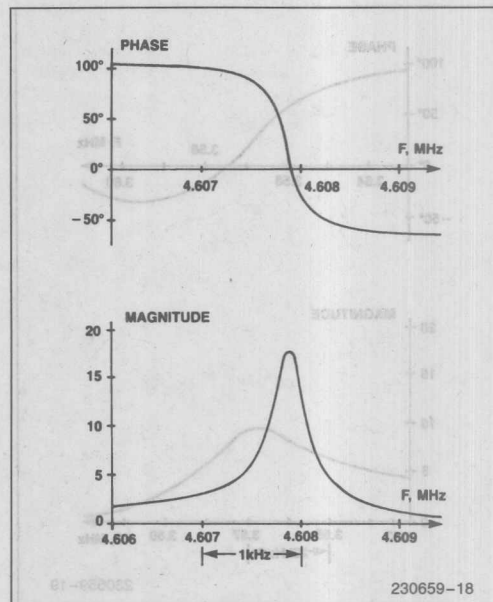


Figure 14. Loop Gain versus Frequency (4.608 MHz Crystal)

The gain of the feedback network is

$$\beta = \frac{Z_i}{Z_i + Z_f}$$

And the loop gain is

$$\beta A = \frac{Z_i}{Z_i + Z_f} \times \frac{A_v Z_L}{Z_L + R_o}$$

The impedances Z_L , Z_f , and Z_i are defined in Figure 13B.

Figure 14 shows the way the loop gain thus calculated (using typical 8051-type parameters and a 4.608 MHz crystal) varies with frequency. The frequency of interest is the one for which the phase of the loop gain is zero. The accepted criterion for start-up is that the magnitude of the loop gain must exceed unity at this frequency. This is the frequency at which the circuit is in resonance. It corresponds very closely with the antiresonant frequency of the motional arm of the crystal in parallel with C_L .

Figure 15 shows the way the loop gain varies with frequency when the parameters of a 3.58 MHz ceramic resonator are used in place of a crystal (the amplifier parameters being typical 8051, as in Figure 14). Note the different frequency scales.

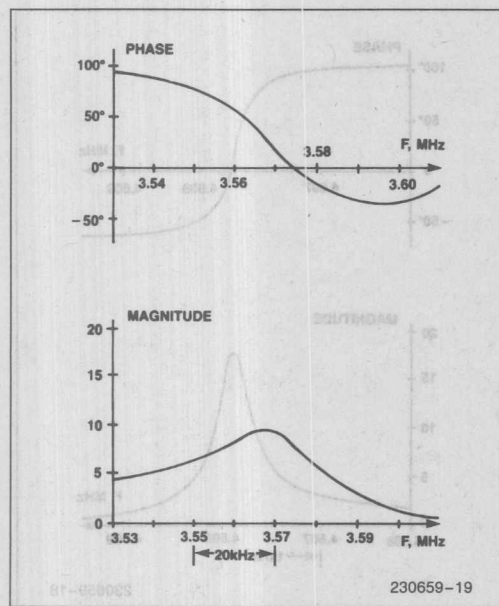


Figure 15. Loop Gain versus Frequency (3.58 MHz Ceramic)

Start-Up Characteristics

It is common, in studies of feedback systems, to examine the behavior of the closed loop gain as a function of complex frequency $s = \sigma + j\omega$; specifically, to determine the location of its poles in the complex plane. A pole is a point on the complex plane where the gain function goes to infinity. Knowledge of its location can be used to predict the response of the system to an input disturbance.

The way that the response function depends on the location of the poles is shown in Figure 16. Poles in the left-half plane cause the response function to take the form of a damped sinusoid. Poles in the right-half plane cause the response function to take the form of an exponentially growing sinusoid. In general,

$$v(t) \sim e^{at} \sin(\omega t + \theta)$$

where a is the real part of the pole frequency. Thus if the pole is in the right-half plane, a is positive and the sinusoid grows. If the pole is in the left-half plane, a is negative and the sinusoid is damped.

The same type of analysis can usefully be applied to oscillators. In this case, however, rather than trying to ensure that the poles are in the left-half plane, we would seek to ensure that they're in the right-half plane. An exponentially growing sinusoid is exactly what is wanted from an oscillator that has just been powered up.

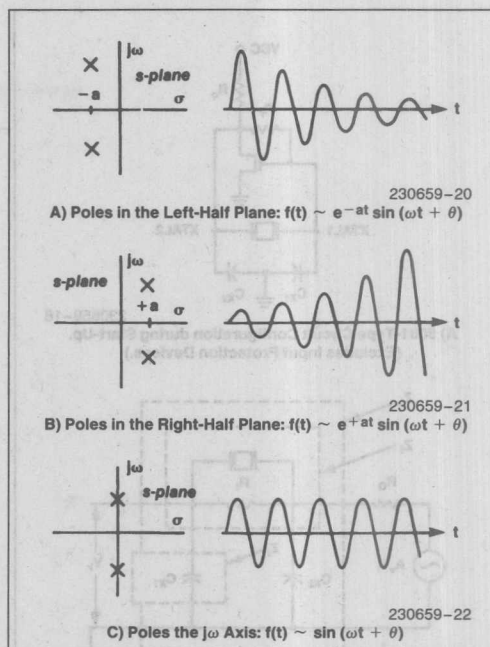


Figure 16. Do You Know Where Your Poles Are Tonight?

The gain function of interest in oscillators is $1/(1 - \beta A)$. Its poles are at the complex frequencies where $\beta A = 1 \angle 0^\circ$, because that value of βA causes the gain function to go to infinity. The oscillator will start up if the real part of the pole frequency is positive. More importantly, the rate at which it starts up is indicated by how much greater than 0 the real part of the pole frequency is.

The circuit in Figure 13B can be used to find the pole frequencies of the oscillator gain function. All that needs to be done is evaluate the impedances at complex frequencies $\sigma + j\omega$ rather than just at ω , and find the value of $\sigma + j\omega$ for which $\beta A = 1 \angle 0^\circ$. The larger that value of σ is, the faster the oscillator will start up.

Of course, other things besides pole frequencies, things like the VCC rise time, are at work in determining the start-up time. But to the extent that the pole frequencies do affect start-up time, we can obtain results like those in Figures 17 and 18.

To obtain these figures the pole frequencies were computed for various values of capacitance C_X from XTAL1 and XTAL2 to ground (thus $C_{X1} = C_{X2} = C_X$). Then a "time constant" for start-up was calculated as $T_s = \frac{1}{\sigma}$ where σ is the real part of the pole frequency (rad/sec), and this time constant is plotted versus C_X .

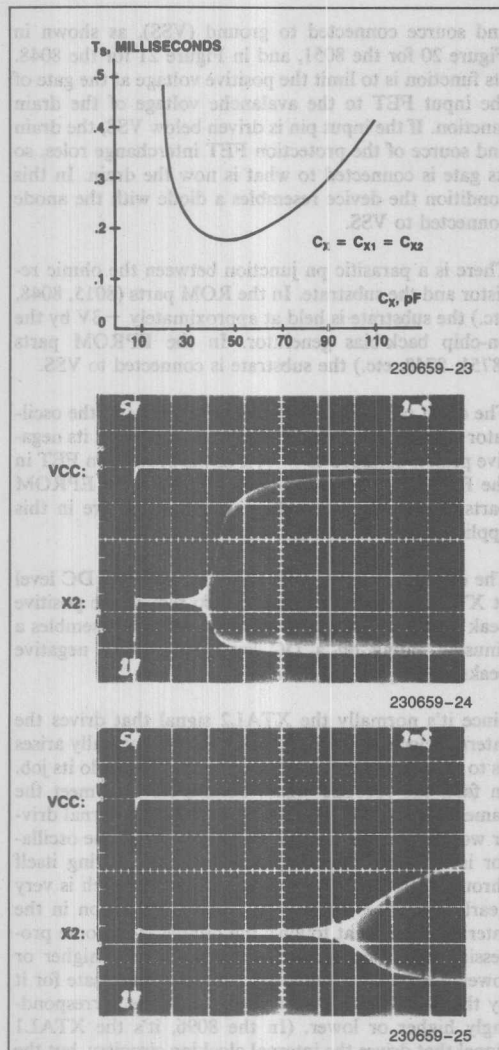


Figure 17. Oscillator Start-Up (4.608 MHz Crystal from Standard Crystal Corp.)

A short time constant means faster start-up. A long time constant means slow start-up. Observations of actual start-ups are shown in the figures. Figure 17 is for a typical 8051 with a 4.608 MHz crystal supplied by Standard Crystal Corp., and Figure 18 is for a typical 8051 with a 3.58 MHz ceramic resonator supplied by NTK Technical Ceramics, Ltd.

It can be seen in Figure 17 that, for this crystal, values of C_x between 30 and 50 pF minimize start-up time, but that the exact value in this range is not particularly important, even if the start-up time itself is critical.

As previously mentioned, start-up time can be taken as an indication of start-up reliability. Start-up problems are normally associated with C_{X1} and C_{X2} being too small or too large for a given resonator. If the parameters of the resonator are known, curves such as in Figure 17 or 18 can be generated to define acceptable ranges of values for these capacitors.

As the oscillations grow in amplitude, they reach a level at which they undergo severe clipping within the amplifier, in effect reducing the amplifier gain. As the amplifier gain decreases, the poles move towards the $j\omega$ axis. In steady-state, the poles are on the $j\omega$ axis and the amplitude of the oscillations is constant.

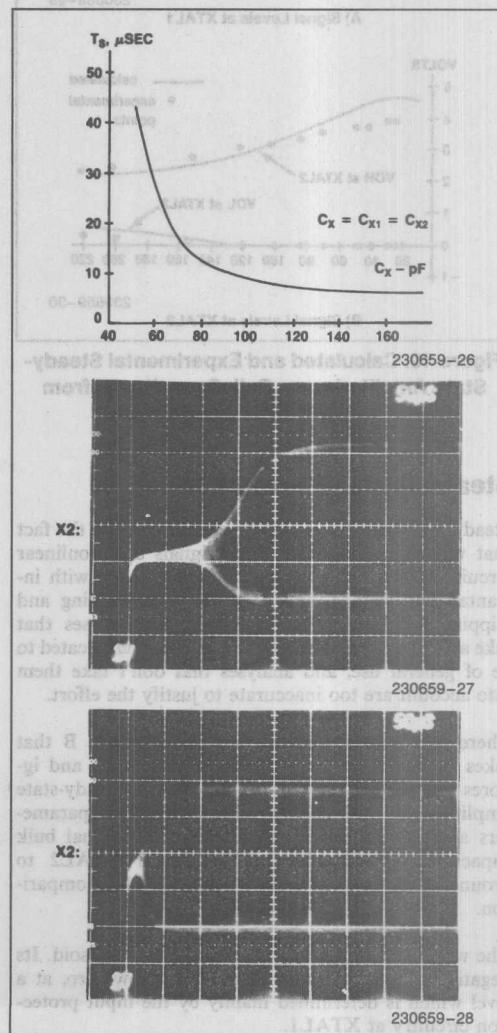


Figure 18. Oscillator Start-Up (3.58 MHz Ceramic Resonator from NTK Technical Ceramics)

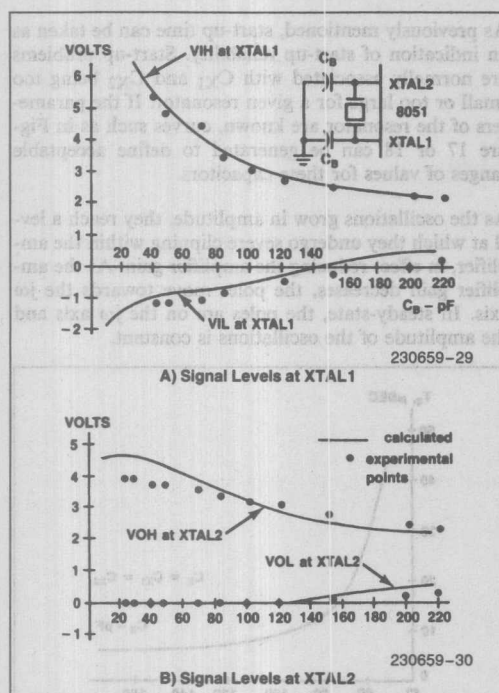


Figure 19. Calculated and Experimental Steady-State Amplitudes vs. Bulk Capacitance from XTAL1 and XTAL2 to Ground

Steady-State Characteristics

Steady-state analysis is greatly complicated by the fact that we are dealing with large signals and nonlinear circuit response. The circuit parameters vary with instantaneous voltage, and a number of clamping and clipping mechanisms come into play. Analyses that take all these things into account are too complicated to be of general use, and analyses that don't take them into account are too inaccurate to justify the effort.

There is a steady-state analysis in Appendix B that takes some of the complications into account and ignores others. Figure 19 shows the way the steady-state amplitudes thus calculated (using typical 8051 parameters and a 4.608 MHz crystal) vary with equal bulk capacitance placed from XTAL1 and XTAL2 to ground. Experimental results are shown for comparison.

The waveform at XTAL1 is a fairly clean sinusoid. Its negative peak is normally somewhat below zero, at a level which is determined mainly by the input protection circuitry at XTAL1.

The input protection circuitry consists of an ohmic resistor and an enhancement-mode FET with the gate

and source connected to ground (VSS), as shown in Figure 20 for the 8051, and in Figure 21 for the 8048. Its function is to limit the positive voltage at the gate of the input FET to the avalanche voltage of the drain junction. If the input pin is driven below VSS, the drain and source of the protection FET interchange roles, so its gate is connected to what is now the drain. In this condition the device resembles a diode with the anode connected to VSS.

There is a parasitic pn junction between the ohmic resistor and the substrate. In the ROM parts (8015, 8048, etc.) the substrate is held at approximately $-3V$ by the on-chip back-bias generator. In the EPROM parts (8751, 8748, etc.) the substrate is connected to VSS.

The effect of the input protection circuitry on the oscillator is that if the XTAL1 signal goes negative, its negative peak is clamped to $-V_{DS}$ of the protection FET in the ROM parts, and to about $-0.5V$ in the EPROM parts. These negative voltages on XTAL1 are in this application self-limiting and nondestructive.

The clamping action does, however, raise the DC level at XTAL1, which in turn tends to reduce the positive peak at XTAL2. The waveform at XTAL2 resembles a sinusoid riding on a DC level, and whose negative peaks are clipped off at zero.

Since it's normally the XTAL2 signal that drives the internal clocking circuitry, the question naturally arises as to how large this signal must be to reliably do its job. In fact, the XTAL2 signal doesn't have to meet the same VIH and VIL specifications that an external driver would have to. That's because as long as the oscillator is working, the on-chip amplifier is driving itself through its own 0-to-1 transition region, which is very nearly the same as the 0-to-1 transition region in the internal buffer that follows the oscillator. If some processing variations move the transition level higher or lower, the on-chip amplifier tends to compensate for it by the fact that its own transition level is correspondingly higher or lower. (In the 8096, it's the XTAL1 signal that drives the internal clocking circuitry, but the same concept applies.)

The main concern about the XTAL2 signal amplitude is an indication of the general health of the oscillator. An amplitude of less than about 2.5V peak-to-peak indicates that start-up problems could develop in some units (with low gain) with some crystals (with high R_1). The remedy is to either adjust the values of C_{X1} and/or C_{X2} or use a crystal with a lower R_1 .

The amplitudes at XTAL1 and XTAL2 can be adjusted by changing the ratio of the capacitors from XTAL1 and XTAL2 to ground. Increasing the XTAL2 capacitance, for example, decreases the amplitude at XTAL2 and increases the amplitude at XTAL1 by about the same amount. Decreasing both caps increases both amplitudes.

Pin Capacitance

Internal pin-to-ground and pin-to-pin capacitances at XTAL1 and XTAL2 will have some effect on the oscillator. These capacitances are normally taken to be in the range of 5 to 10 pF, but they are extremely difficult to evaluate. Any measurement of one such capacitance will necessarily include effects from the others. One advantage of the positive reactance oscillator is that the pin-to-ground capacitances are paralleled by external bulk capacitors, so a precise determination of their value is unnecessary. We would suggest that there is little justification for more precision than to assign them a value of 7 pF (XTAL1-to-ground and XTAL1-to-XTAL2). This value is probably not in error by more than 3 or 4 pF.

The XTAL2-to-ground capacitance is not entirely "pin capacitance," but more like an "equivalent output capacitance" of some 25 to 30 pF, having to include the effect of internal phase delays. This value will vary to some extent with temperature, processing, and frequency.

MCS®-51 Oscillator

The on-chip amplifier on the HMOS MCS-51 family is shown in Figure 20. The drain load and feedback "resistors" are seen to be field-effect transistors. The drain load FET, R_D , is typically equivalent to about 1K to 3 K-ohms. As an amplifier, the low frequency voltage gain is normally between -10 and -20, and the output resistance is effectively R_D .

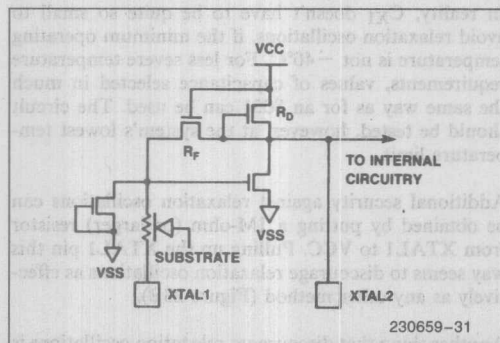


Figure 20. MCS®-51 Oscillator Amplifier

The 80151 oscillator is normally used with equal bulk capacitors placed externally from XTAL1 to ground and from XTAL2 to ground. To determine a reasonable value of capacitance to use in these positions, given a crystal of ceramic resonator of known parameters, one can use the BASIC analysis in Appendix II to generate curves such as in Figures 17 and 18. This procedure will define a range of values that will minimize start-up time. We don't suggest that smaller values be

used than those which minimize start-up time. Larger values than those can be used in applications where increased frequency stability is desired, at some sacrifice in start-up time.

Standard Crystal Corp. (Reference 8) studied the use of their crystals with the MCS-51 family using skew sample supplied by Intel. They suggest putting 30 pF capacitors from XTAL1 and XTAL2 to ground, if the crystal is specified as described in Reference 8. They noted that in that configuration and with crystals thus specified, the frequency accuracy was $\pm 0.01\%$ and the frequency stability was $\pm 0.005\%$, and that a frequency accuracy of $\pm 0.005\%$ could be obtained by substituting a 25 pF fixed cap in parallel with a 5-20 pF trimmer for one of the 30 pF caps.

MCS-51 skew samples have also been supplied to a number of ceramic resonator manufacturers for characterization with their products. These companies should be contacted for application information on their products. In general, however, ceramics tend to want somewhat larger values for C_{X1} and C_{X2} than quartz crystals do. As shown in Figure 18, they start up a lot faster that way.

In some application the actual frequency tolerance required is only 1% or so, the user being concerned mainly that the circuit will oscillate. In that case, C_{X1} and C_{X2} can be selected rather freely in the range of 20 to 80 pF.

As you can see, "best" values for these components and their tolerances are strongly dependent on the application and its requirements. In any case, their suitability should be verified by environmental testing before the design is submitted to production.

MCS®-48 Oscillator

The NMOS and HMOS MCS-48 oscillator is shown in Figure 21. It differs from the 8051 in that its inverting

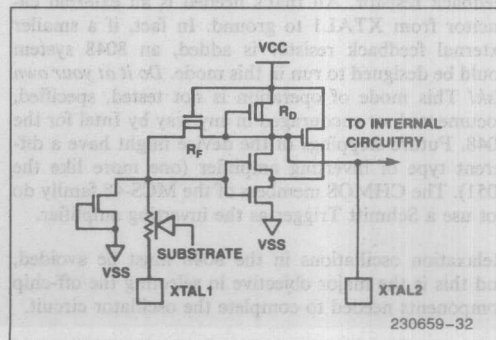


Figure 21. MCS®-48 Oscillator Amplifier

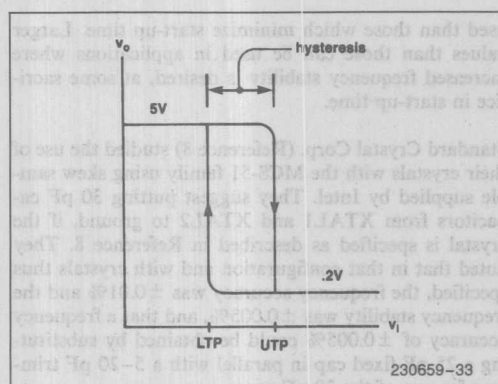


Figure 22. Schmitt Trigger Characteristic

amplifier is a Schmitt Trigger. This configuration was chosen to prevent crosstalk from the TO pin, which is adjacent to the XTAL1 pin.

All Schmitt Trigger circuits exhibit a hysteresis effect, as shown in Figure 22. The hysteresis is what makes it less sensitive to noise. The same hysteresis allows any Schmitt Trigger to be used as a relaxation oscillator. All you have to do is connect a resistor from output to input, and a capacitor from input to ground, and the circuit oscillates in a relaxation mode as follows.

If the Schmitt Trigger output is at a logic high, the capacitor commences charging through the feedback resistor. When the capacitor voltage reaches the upper trigger point (UTP), the Schmitt Trigger output switches to a logic low and the capacitor commences discharging through the same resistor. When the capacitor voltage reaches the lower trigger point (LTP), the Schmitt Trigger output switches to a logic high again, and the sequence repeats. The oscillation frequency is determined by the RC time constant and the hysteresis voltage, UTP-LTP.

The 8048 can oscillate in this mode. It has an internal feedback resistor. All that's needed is an external capacitor from XTAL1 to ground. In fact, if a smaller external feedback resistor is added, an 8048 system could be designed to run in this mode. *Do it at your own risk!* This mode of operation is not tested, specified, documented, or encouraged in any way by Intel for the 8048. Future steppings of the device might have a different type of inverting amplifier (one more like the 8051). The CHMOS members of the MCS-48 family do not use a Schmitt Trigger as the inverting amplifier.

Relaxation oscillations in the 8048 must be avoided, and this is the major objective in selecting the off-chip components needed to complete the oscillator circuit.

When an 8048 is powered up, if VCC has a short rise time, the relaxation mode starts first. The frequency is normally about 50 KHz. The resonator mode builds

more slowly, but it eventually takes over and dominates the operation of the circuit. This is shown in Figure 23A.

Due to processing variations, some units seem to have a harder time coming out of the relaxation mode, particularly at low temperatures. In some cases the resonator oscillations may fail entirely, and leave the device in the relaxation mode at any temperature if C_{X1} is larger than about 50 pF. Therefore, C_{X1} should be chosen with some care, particularly if the system must operate at lower temperatures.

One method that has proven effective in all units to -40°C is to put 5 pF from XTAL1 to ground and 20 pF from XTAL2 to ground. Unfortunately, while this method does discourage the relaxation mode, it is not an optimal choice for the resonator mode. For one thing, it does not swamp the pin capacitance. Also, it makes for a rather high signal level at XTAL1 (8 or 9 volts peak-to-peak).

The question arises as to whether that level of signal at XTAL1 might damage the chip. Not to worry. The negative peaks are self-limiting and nondestructive. The positive peaks could conceivably damage the oxide, but in fact, NMOS chips (eg, 8048) and HMOS chips (eg, 8048H) are tested to a much higher voltage than that. The technology trend, of course, is to thinner oxides, as the devices shrink in size. For an extra margin of safety, the HMOS II chips (eg, 8048AH) have an internal diode clamp at XTAL1 to VCC.

In reality, C_{X1} doesn't have to be quite so small to avoid relaxation oscillations, if the minimum operating temperature is not -40°C . For less severe temperature requirements, values of capacitance selected in much the same way as for an 8051 can be used. The circuit should be tested, however, at the system's lowest temperature limit.

Additional security against relaxation oscillations can be obtained by putting a 1M-ohm (or larger) resistor from XTAL1 to VCC. Pulling up the XTAL1 pin this way seems to discourage relaxation oscillations as effectively as any other method (Figure 23B).

Another thing that discourages relaxation oscillations is low VCC. The resonator mode, on the other hand is much less sensitive to VCC. Thus if VCC comes up relatively slowly (several milliseconds rise time), the resonator mode is normally up and running before the relaxation mode starts (in fact, before VCC has even reached operating specs). This is shown in Figure 23C.

A secondary effect of the hysteresis is a shift in the oscillation frequency. At low frequencies, the output signal from an inverter without hysteresis leads (or lags) the input by 180 degrees. The hysteresis in a Schmitt Trigger, however, causes the output to lead the

input by less than 180 degrees (or lag by more than 180 degrees), by an amount that depends on the signal amplitude, as shown in Figure 24. At higher frequencies, there are additional phase shifts due to the various reactances in the circuit, but the phase shift due to the hysteresis is still present. Since the total phase shift in the oscillator's loop gain is necessarily 0 or 360 degrees, it is apparent that as the oscillations build up, the frequency has to change to allow the reactances to compensate for the hysteresis. In normal operation, this additional phase shift due to hysteresis does not exceed a few degrees, and the resulting frequency shift is negligible.

Kyocera, a ceramic resonator manufacturer, studied the use of some of their resonators (at 6.0 MHz, 8.0 MHz, and 11.0 MHz) with the 8049H. Their conclusion as to the value of capacitance to use at XTAL1 and XTAL2 was that 33 pF is appropriate at all three frequencies. One should probably follow the manufacturer's recommendations in this matter, since they will guarantee operation.

Whether one should accept these recommendations and guarantees without further testing is, however, another matter. Not all users have found the recommendations to be without occasional problems. If you run into diffi-

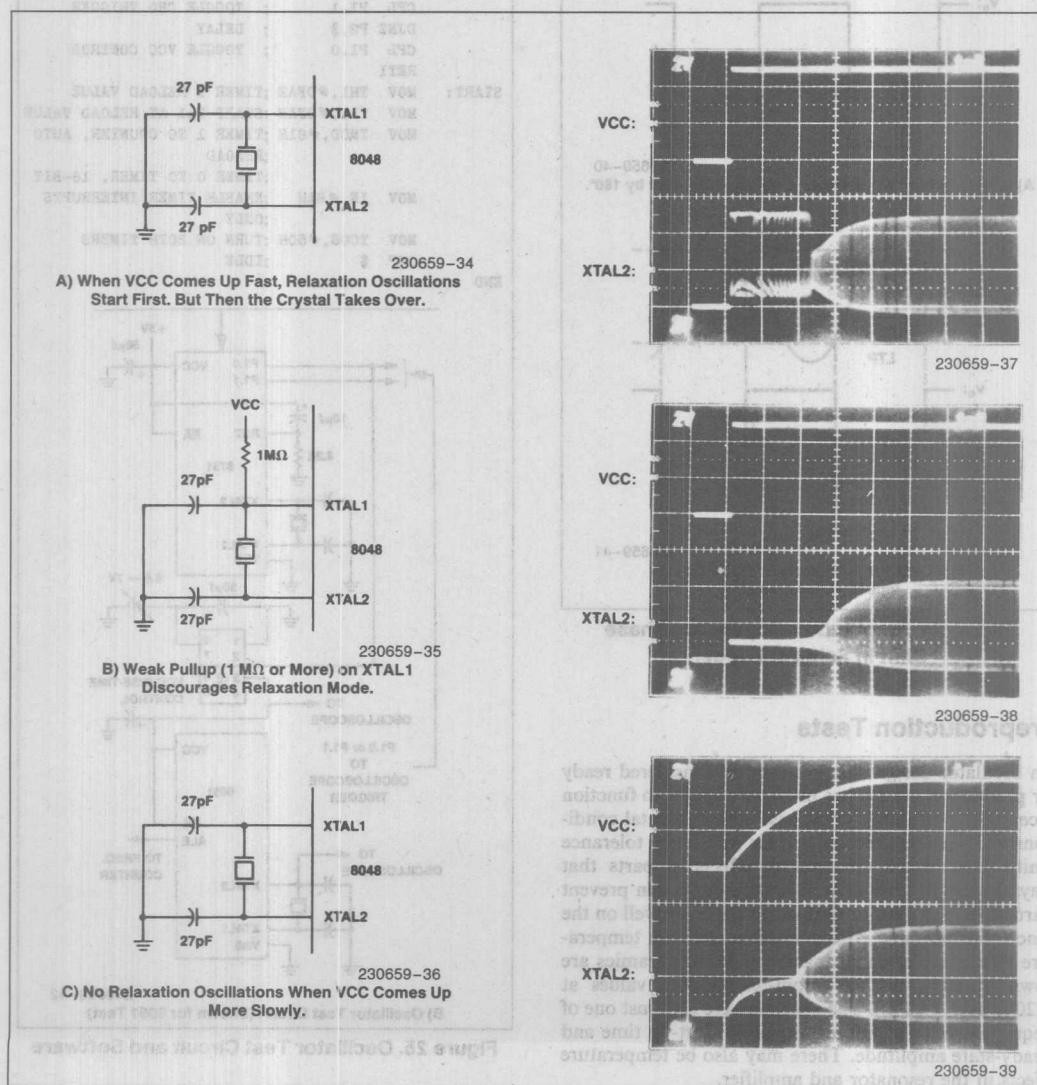


Figure 23. Relaxation Oscillations in the 8048

culties using their recommendations, both Intel and the ceramic resonator manufacturer want to know about it. It is to their interest, and ours, that such problems be resolved.

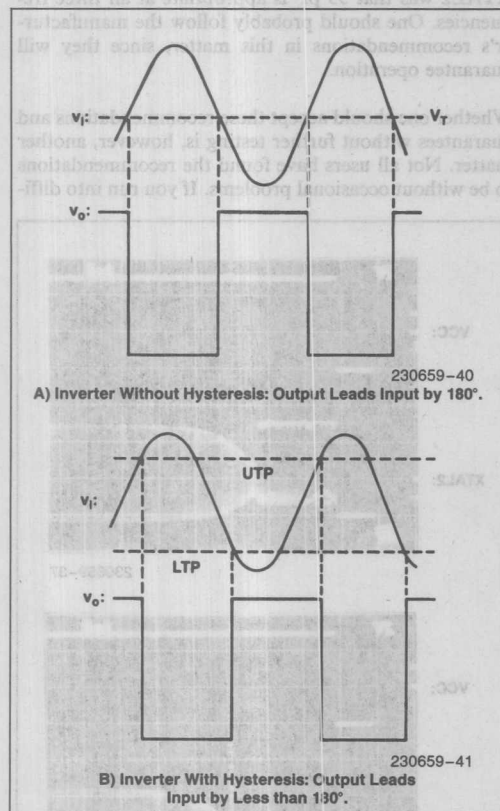


Figure 24. Amplitude-Dependent Phase Shift in Schmitt Trigger

Preproduction Tests

An oscillator design should never be considered ready for production until it has proven its ability to function acceptably well under worst-case environmental conditions and with parameters at their worst-case tolerance limits. Unexpected temperature effects in parts that may already be near their tolerance limits can prevent start-up of an oscillator that works perfectly well on the bench. For example, designers often overlook temperature effects in ceramic capacitors. (Some ceramics are down to 50% of their room-temperature values at -20°C and $+60^{\circ}\text{C}$). The problem here isn't just one of frequency stability, but also involves start-up time and steady-state amplitude. There may also be temperature effects in the resonator and amplifier.

It will be helpful to build a test jig that will allow the oscillator circuit to be tested independently of the rest of the system. Both start-up and steady-state characteristics should be tested. Figure 25 shows the circuit that

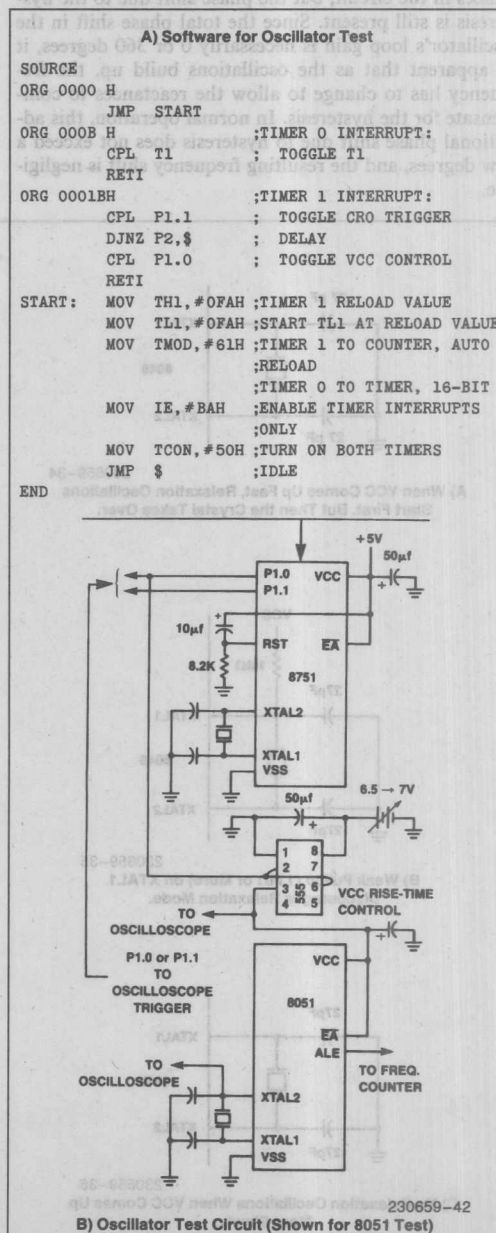


Figure 25. Oscillator Test Circuit and Software

was used to obtain the oscillator start-up photographs in this Application Note. This circuit or a modified version of it would make a convenient test vehicle. The oscillator and its relevant components can be physically separated from the control circuitry, and placed in a temperature chamber.

Start-up should be observed under a variety of conditions, including low VCC and using slow and fast VCC rise times. The oscillator should not be reluctant to start up even when VCC is below its spec value for the rest of the chip. (The rest of the chip may not function, but the oscillator should work.) It should also be verified that start-up occurs when the resonator has more than its upper tolerance limit of series resistance. (Put some resistance in series with the resonator for this test.) The bulk capacitors from XTAL1 and XTAL2 to ground should also be varied to their tolerance limits.

The same circuit, with appropriate changes in the software to lengthen the "on" time, can be used to test the steady-state characteristics of the oscillator, specifically the frequency, frequency stability, and amplitudes at XTAL1 and XTAL2.

As previously noted, the voltage swings at these pins are not critical, but they should be checked at the system's temperature limits to ensure that they are in good health. Observing these signals necessarily changes them somewhat. Observing the signal at XTAL2 requires that the capacitor at that pin be reduced to account for the oscilloscope probe capacitance. Observing the signal at XTAL1 requires the same consideration, plus a blocking capacitor (switch the oscilloscope input to AC), so as to not disturb the DC level at that pin. Alternatively, a MOSFET buffer such as the one shown in Figure 26 can be used. It should be verified by direct measurement that the ground clip on the scope probe is ohmically connected to the scope chassis (probes are incredibly fragile in this respect), and the observations should be made with the ground clip on the VSS pin, or very close to it. If the probe shield isn't operational and in use, the observations are worthless.

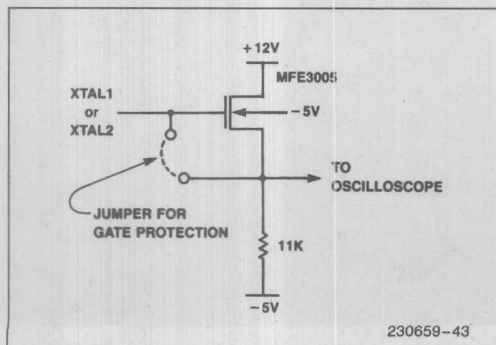


Figure 26. MOSFET Buffer for Observing Oscillator Signals

Frequency checks should be made with only the oscillator circuitry connected to XTAL1 and XTAL2. The ALE frequency can be counted, and the oscillator frequency derived from that. In systems where the frequency tolerance is only "nominal," the frequency should still be checked to ascertain that the oscillator isn't running in a spurious resonance or relaxation mode. Switching VCC off and on again repeatedly will help reveal a tendency to go into unwanted modes of oscillation.

The operation of the oscillator should then be verified under actual system running conditions. By this stage one will be able to have some confidence that the basic selection of components for the oscillator itself is suitable, so if the oscillator appears to malfunction in the system the fault is not in the selection of these components.

Troubleshooting Oscillator Problems

The first thing to consider in case of difficulty is that between the test jig and the actual application there may be significant differences in stray capacitances, particularly if the actual application is on a multi-layer board.

Noise glitches, that aren't present in the test jig but are in the application board, are another possibility. Capacitive coupling between the oscillator circuitry and other signal has already been mentioned as a source of miscounts in the internal clocking circuitry. Inductive coupling is also possible, if there are strong currents nearby. These problems are a function of the PCB layout.

Surrounding the oscillator components with "quiet" traces (VCC and ground, for example) will alleviate capacitive coupling to signals that have fast transition times. To minimize inductive coupling, the PCB layout should minimize the areas of the loops formed by the oscillator components. These are the loops that should be checked:

- XTAL1 through the resonator to XTAL2;
- XTAL1 through C_{X1} to the VSS pin;
- XTAL2 through C_{X2} to the VSS pin.

It is not unusual to find that the grounded ends of C_{X1} and C_{X2} eventually connect up to the VSS pin only after looping around the farthest ends of the board. Not good.

Finally, it should not be overlooked that software problems sometimes imitate the symptoms of a slow-starting oscillator or incorrect frequency. Never underestimate the perversity of a software problem.

REFERENCES

1. Frerking, M. E., *Crystal Oscillator Design and Temperature Compensation*, Van Nostrand Reinhold, 1978.
2. Bottom, V., "The Crystal Unit as a Circuit Component," Ch. 7, *Introduction to Quartz Crystal Unit Design*, Van Nostrand Reinhold, 1982.
3. Parzen, B., *Design of Crystal and Other Harmonic Oscillators*, John Wiley & Sons, 1983.
4. Holmbeck, J. D., "Frequency Tolerance Limitations with Logic Gate Clock Oscillators, 31st Annual Frequency Control Symposium, June, 1977.
5. Roberge, J. K., "Nonlinear Systems," Ch. 6, *Operational Amplifiers: Theory and Practice*, Wiley, 1975.
6. Eaton, S. S., *Timekeeping Advances Through COS/MOS Technology*, RCA Application Note ICAN-6086.
7. Eaton, S. S., *Micropower Crystal-Controlled Oscillator Design Using RCA COS/MOS Inverters*, RCA Application Note ICAN-6539.
8. Fisher, J. B., *Crystal Specifications for the Intel 8031/8051/8751 Microcontrollers*, Standard Crystal Corp. Design Data Note #2F.
9. Murata Mfg. Co., Ltd., *Ceramic Resonator "Ceralock" Application Manual*.
10. Kyoto Ceramic Co., Ltd., *Adaptability Test Between Intel 8049H and Kyocera Ceramic Resonators*.
11. Kyoto Ceramic Co., Ltd., *Technical Data on Ceramic Resonator Model KBR-6.0M, KBR-8.0M, KBR-11.0M Application for 8051 (Intel)*.
12. NTK Technical Ceramic Division, NGK Spark Plug Co., Ltd., *NTKK Ceramic Resonator Manual*.

As previously noted, the voltage swing at these pins is not critical, but they should be checked at the system temperature limits to ensure that they are in good health. Observing these signals necessarily changes their waveform. Observing the signal at XTAL2 requires that the capacitor at that pin be reduced to account for the oscilloscope probe capacitance. Observing the signal at XTAL1 requires the same consideration, plus a pickoff capacitor (which the oscilloscope input pin to AC), so as to not disturb the DC level at that pin. Alternatively, a MOSFET buffer such as the one shown in Figure 28 can be used. It should be verified by direct measurement that the ground clip on the scope probe is electrically connected to the scope chassis (probes are incredibly fragile in this respect), and the observations should be made with the ground clip on the VSS pin, or very close to it. If the probe shield isn't operational and in use, the observations are worthless.

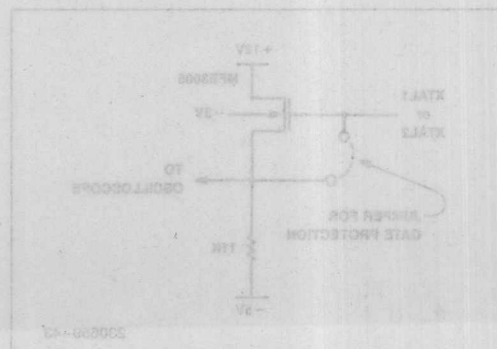


Figure 28. MOSFET Buffer for Observing Oscillator Signals

XTAL1 through the resonator to XTAL2.
XTAL1 through C₁ to the VSS pin.
XTAL2 through C₂ to the VSS pin.

It is not unusual to find that the grounded ends of C₁ and C₂ eventually connect up to the VSS pin only after looping around the farthest end of the board. Not good.

Finally, it should not be overlooked that software problems sometimes imitate the symptoms of a low-quality oscillator or incorrect frequency. Never underestimate the perversity of a software problem.

APPENDIX A QUARTZ AND CERAMIC RESONATOR FORMULAS

Based on the equivalent circuit of the crystal, the impedance of the crystal is

$$Z_{XTAL} = \frac{(R_1 + j\omega L_1 + 1/j\omega C_1)(1/j\omega C_0)}{R_1 + j\omega L_1 + 1/j\omega C_1 + 1/j\omega C_0}$$

After some algebraic manipulation, this calculation can be written in the form

$$Z_{XTAL} = \frac{1}{j\omega(C_1 + C_0)} \cdot \frac{1 - \omega^2 L_1 C_1 + j\omega R_1 C_1}{1 - \omega^2 L_1 C_T + j\omega R_1 C_T}$$

where C_T is the capacitance of C_1 in series with C_0 :

$$C_T = \frac{C_1 C_0}{C_1 + C_0}$$

The impedance of the crystal in parallel with an external load capacitance C_L is the same expression, but with $C_0 + C_L$ substituted for C_0 :

$$Z_{XTAL \parallel CL} = \frac{1}{j\omega(C_1 + C_0 + C_L)} \cdot \frac{1 - \omega^2 L_1 C_1 + j\omega R_1 C_1}{1 - \omega^2 L_1 C_T + j\omega R_1 C_T}$$

where C_T is the capacitance of C_1 in series with $(C_0 + C_L)$:

$$C_T = \frac{C_1(C_0 + C_L)}{C_1 + C_0 + C_L}$$

The impedance of the crystal in series with the load capacitance is

$$Z_{XTAL + CL} = Z_{XTAL} + \frac{1}{j\omega C_L} = \frac{C_L + C_1 + C_0}{j\omega C_L(C_1 + C_0)} \cdot \frac{1 - \omega^2 L_1 C_T + j\omega R_1 C_T}{1 - \omega^2 L_1 C_T + j\omega R_1 C_T}$$

where C_T and C'_T are as defined above.

The phase angles of these impedances are readily obtained from the impedance expressions themselves:

$$\theta_{XTAL} = \arctan \frac{\omega R_1 C_1}{1 - \omega^2 L_1 C_1} - \arctan \frac{\omega R_1 C_T}{1 - \omega^2 L_1 C_T} - \frac{\pi}{2}$$

$$\theta_{XTAL \parallel CL} = \arctan \frac{\omega R_1 C_1}{1 - \omega^2 L_1 C_1} - \arctan \frac{\omega R_1 C'_T}{1 - \omega^2 L_1 C'_T} - \frac{\pi}{2}$$

$$\theta_{XTAL + CL} = \arctan \frac{\omega R_1 C'_T}{1 - \omega^2 L_1 C'_T} - \arctan \frac{\omega R_1 C_T}{1 - \omega^2 L_1 C_T} - \frac{\pi}{2}$$

The resonant ("series resonant") frequency is the frequency at which the phase angle is zero and the impedance is low. The antiresonant ("parallel resonant") frequency is the frequency at which the phase angle is zero and the impedance is high.

Each of the above θ -expressions contains two arctan functions. Setting the denominator of the argument of the first arctan function to zero gives (approximately) the "series resonant" frequency for that configuration. Setting the denominator of the argument of the second arctan function to zero gives (approximately) the "parallel resonant" frequency for that configuration.

For example, the resonant frequency of the crystal is the frequency at which

$$1 - \omega^2 L_1 C_1 = 0$$

Thus

$$\omega_s = \frac{1}{\sqrt{L_1 C_1}}$$

or

$$f_s = \frac{1}{2\pi\sqrt{L_1 C_1}}$$

It will be noted that the series resonant frequency of the "XTAL + CL" configuration (crystal in series with CL) is the same as the parallel resonant frequency of the "XTAL || CL" configuration (crystal in parallel with CL). This is the frequency at which

$$1 - \omega^2 L_1 C_T = 0$$

Thus

$$\omega_a = \frac{1}{\sqrt{L_1 C_T}}$$

or

$$f_a = \frac{1}{2\pi\sqrt{L_1 C_T}}$$

This fact is used by crystal manufacturers in the process of calibrating a crystal to a specified load capacitance.

By subtracting the resonant frequency of the crystal from its antiresonant frequency, one can calculate the range of frequencies over which the crystal reactance is positive:

$$f_a - f_s = f_s \sqrt{1 + C_1/C_0} - 1$$

$$f_s \left(\frac{C_1}{2C_0} \right)$$

Given typical values for C_1 and C_0 , this range can hardly exceed 0.5% of f_s . Unless the inverting amplifier in the positive reactance oscillator is doing something very strange indeed, the oscillation frequency is bound to be accurate to that percentage whether the crystal was calibrated for series operation or to any unspecified load capacitance.

Equivalent Series Resistance

ESR is the real part of Z_{XTAL} at the oscillation frequency. The oscillation frequency is the parallel resonant frequency of the "XTAL || CL" configuration (which is the same as the series resonant frequency of the "XTAL + CL" configuration). Substituting this frequency into the Z_{XTAL} expression yields, after some algebraic manipulation,

$$ESR = \frac{R_1 \left(\frac{C_0 + C_L}{C_L} \right)^2}{1 + \omega^2 C_1^2 \left(\frac{C_0 + C_L}{C_L} \right)^2} \approx R_1 \left(1 + \frac{C_0}{C_L} \right)^2$$

Drive Level

The power dissipated by the crystal is $I_1^2 R_1$, where I_1 is the RMS current in the motional arm of the crystal. This current is given by $V_x / |Z_1|$, where V_x is the RMS voltage across the crystal, and $|Z_1|$ is the magnitude of the impedance of the motional arm. At the oscillation frequency, the motional arm is a positive (inductive) reactance in parallel resonance with $(C_0 + C_L)$. Therefore $|Z_1|$ is approximately equal to the magnitude of the reactance of $(C_0 + C_L)$:

$$|Z_1| = \frac{1}{2\pi f(C_0 + C_L)}$$

where f is the oscillation frequency. Then,

$$P = I_1^2 R_1 = \left(\frac{V_x}{|Z_1|} \right)^2 R_1$$

$$= [2\pi f(C_0 + C_L) V_x]^2 R_1$$

The waveform of the voltage across the crystal (XTAL1 to XTAL2) is approximately sinusoidal. If its peak value is V_{CC} , then V_x is $V_{CC}/\sqrt{2}$. Therefore,

$$P = 2R_1 [\pi f(C_0 + C_L) V_{CC}]^2$$

APPENDIX B OSCILLATOR ANALYSIS PROGRAM

The program is written in BASIC. BASIC is excruciatingly slow, but it has some advantages. For one thing, more people know BASIC than FORTRAN. In addition, a BASIC program is easy to develop, modify, and "fiddle around" with. Another important advantage is that a BASIC program can run on practically any small computer system.

Its slowness is a problem, however. For example, the routine which calculates the "start-up time constant" discussed in the text may take several hours to complete. A person who finds this program useful may prefer to convert it to FORTAN, if the facilities are available.

Limitations of the Program

The program was developed with specific reference to 8051-type oscillator circuitry. That means the on-chip amplifier is a simple inverter, and not a Schmitt Trigger. The 8096, the 80C51, the 80C48 and 80C49 all have simple inverters. The 8096 oscillator is almost identical to the 8051, differing mainly in the input protection circuitry. The CHMOS amplifiers have somewhat different parameters (higher gain, for example), and different transition levels than the 8051.

The MCS-48 family is specifically included in the program only to the extent that the input-output curve used in the steady-state analysis is that of a Schmitt Trigger, if the user identifies the device under analysis as an MCS-48 device. The analysis does not include the voltage dependent phase shift of the Schmitt Trigger.

The clamping action of the input protection circuitry is important in determining the steady-state amplitudes. The steady-state routine accounts for it by setting the negative peak of the XTAL1 signal at a level which depends on the amplitude of the XTAL1 signal in accordance with experimental observations. It's an exercise in curve-fitting. A user may find a different type of curve works better. Later steppings of the chips may behave differently in this respect, having somewhat different types of input protection circuitry.

It should be noted that the analysis ignores a number of important items, such as high-frequency effects in the on-chip circuitry. These effects are difficult to predict, and are no doubt dependent on temperature, frequency, and device sample. However, they can be simulated to a reasonable degree by adding an "output capacitance" of about 20 pF to the circuit model (i.e., in parallel with CX2) as described below.

Notes on Using the Program

The program asks the user to input values for various circuit parameters. First the crystal (or ceramic resonator) parameters are asked for. These are R1, L1, C1, and C0. The manufacturer can supply these values for selected samples. To obtain any kind of correlation between calculation and experiment, the values of these parameters must be known for the specific sample in the test circuit. The value that should be entered for C0 is the C0 of the crystal itself plus an estimated 7 pF to account for the XTAL1-to-XTAL2 pin capacitance, plus any other stray capacitance paralleling the crystal that the user may feel is significant enough to be included.

Then the program asks for the values of the XTAL1-to-ground and XTAL2-to-ground capacitances. For CXTAL1, enter the value of the externally connected bulk capacitor plus an estimated 7 pF for pin capacitance. For CXTAL2, enter the value of the externally connected bulk capacitor plus an estimated 7 pF for pin capacitance plus about 20 pF to simulate high-frequency roll-off and phase shifts in the on-chip circuitry.

Next the program asks for values for the small-signal parameters of the on-chip amplifier. Typically, for the 8051/8751,

Amplifier Gain Magnitude = 15
Feedback Resistance = 2300 K Ω
Output Resistance = 2 K Ω

The same values can be used for MCS-48 (NMOS and HMOS) devices, but they are difficult to verify, because the Schmitt Trigger does not lend itself to small-signal measurements.

APRIL 8, 1983

```

100 DEFDBL C,D,F,G,L,P,R,S,X
200 REM
300 REM *****
400 REM
500 REM
600 REM
700 REM
800 REM FNZM(R,X) = MAGNITUDE OF A COMPLEX NUMBER, !R+JX!
900 DEF FNZM(R,X) = SQR(R^2+X^2)
1000 REM
1100 REM FNZP(R,X) = ANGLE OF A COMPLEX NUMBER
1200 REM = 180/PI*ARCTAN(X/R) IF R>0
1300 REM = 180/PI*ARCTAN(X/R) + 180 IF R<0 AND X>0
1400 REM = 180/PI*ARCTAN(X/R) - 180 IF R<0 AND X<0
1500 DEF FNZP(R,X) = 180/PI*ATN(X/R) - ((SGN(R)-1)*SGN(X)*90)
1600 REM
1700 REM INDUCTIVE IMPEDANCE AT COMPLEX FREQUENCY S+JF (HZ)
1800 REM Z = 2*PI*S*L + J2*PI*F*L
1900 REM = FNRL(S,L) + JFNXL(F,L)
2000 DEF FNRL(S,L) = 2*PI*S*L
2100 DEF FNXL(F,L) = 2*PI*F*L
2200 REM
2300 REM CAPACITIVE IMPEDANCE AT COMPLEX FREQUENCY S+JF (HZ)
2400 REM Z = 1/[2*PI*(S+JF)*C]
2500 REM = S/[2*PI*(S^2+F^2)*C] + J(-F)/[2*PI*(S^2+F^2)*C]
2600 REM = FNRC(S,F,C) + JFNXC(S,F,C)
2700 DEF FNRC(S,F,C) = S/(2*PI*(S^2+F^2)*C)
2800 DEF FNXC(S,F,C) = -F/(2*PI*(S^2+F^2)*C)
2900 REM
3000 REM RATIO OF TWO COMPLEX NUMBERS
3100 REM RA+JXA RA+RB+XA*XB XA*RB-RA*XB
3200 REM ----- + J -----
3300 REM RB+JXB RB^2+XB^2 RB^2+XB^2
3400 REM = FNRR(RA,XA,RB,XB) + JFNXR(RA,XA,RB,XB)
3500 DEF FNRR(RA,XA,RB,XB) = (RA*RB+XA*XB)/(RB^2+XB^2)
3600 DEF FNXR(RA,XA,RB,XB) = (XA*RB-RA*XB)/(RB^2+XB^2)
3700 REM
3800 REM PRODUCT OF TWO COMPLEX NUMBERS
3900 REM (RA+JXA)*(RB+JXB) = RA*RB-XA*XB + J(XA*RB+RA*XB)
4000 REM = FNRM(RA,XA,RB,XB) + JFNXM(RA,XA,RB,XB)
4100 DEF FNRM(RA,XA,RB,XB) = RA*RB - XA*XB
4200 DEF FNXM(RA,XA,RB,XB) = XA*RB + RA*XB
4300 REM
4400 REM
4500 REM PARALLEL IMPEDANCES
4600 REM (RA+JXA):(RB+JXB) = (RA+JXA)*(RB+JXB)
4700 REM RA+RB + J(XA+XB)
4800 REM
4900 REM
5000 REM RA*(RB^2+XB^2)+RB*(RA^2+XA^2) XA*(RB^2+XB^2)+XB*(RA^2+XA^2)
5100 REM = ----- + J -----
5200 REM (RA+RB)^2 + (XA+XB)^2 (RA+RB)^2 + (XA+XB)^2
5300 REM
5400 REM = FNRP(RA,XA,RB,XB) + JFNXP(RA,XA,RB,XB)
5500 DEF FNRP(RA,XA,RB,XB) = (RA*(RB^2+XB^2) + RB*(RA^2+XA^2))/((RA+RB)^2 + (XA+XB)^2)
5600 DEF FNXP(RA,XA,RB,XB) = (XA*(RB^2+XB^2) + XB*(RA^2+XA^2))/((RA+RB)^2 + (XA+XB)^2)
5700 REM
5800 REM *****
5900 REM
6000 REM BEGIN COMPUTATIONS
6100 REM
6200 LET PI = 3.141592654#
6300 REM
6400 REM
6500 GOSUB 14300
6600 REM
6700 REM ESTABLISH NOMINAL RESONANT AND ANTIRESONANT CRYSTAL FREQUENCIES
6800 FS = FIX(1/(2*PI*SQR(L1*C1)))
6900 FA = FIX(1/(2*PI*SQR(L1*C1*CO/(C1+CO))))
7000 PRINT
7100 PRINT "XTAL IS SERIES RESONANT AT ",FS," HZ"
7200 PRINT " PARALLEL RESONANT AT ",FA," HZ"
7300 PRINT
7400 PRINT "SELECT: 1. LIST PARAMETERS"
7500 PRINT " 2. CIRCUIT ANALYSIS"
7600 PRINT " 3. OSCILLATION FREQUENCY"
7700 PRINT " 4. START-UP TIME CONSTANT"
7800 PRINT " 5. STEADY-STATE ANALYSIS"

```

230659-44

```

7900 PRINT
8000 INPUT N
8100 IF N=1 THEN PRINT ELSE E600
8200 REM
8300 REM ----- LIST PARAMETERS -----
8400 GOSUB 17100
8500 GOTO 6800
8600 IF N=2 THEN PRINT ELSE 9400
8700 REM
8800 REM ----- CIRCUIT ANALYSIS -----
8900 PRINT " FREQUENCY S+JF TYPE (S), (F) "
9000 INPUT SG,FQ
9100 GOSUB 20200
9200 GOSUB 26600
9300 GOTO 6800
9400 IF N=3 THEN 10300 ELSE 11000
9500 REM
9600 REM ----- OSCILLATION FREQUENCY -----
9700 CL = (CX*CY)/(CX+CY) + CO
9800 FQ = FIX(1/(2*PI*SQR(L1*CL/(C1+CL))))
9900 SQ = 0
10000 DF = FIX(10*INT(LOG(FA-FS)/LOG(10)-2)+5)
10100 DS = 0
10200 RETURN
10300 GOSUB 9700
10400 GOSUB 30300
10500 PRINT
10600 PRINT
10700 PRINT "FREQUENCY AT WHICH LOOP GAIN HAS ZERO PHASE ANGLE:"
10800 GOSUB 26600
10900 GOTO 6800
11000 IF N=4 THEN PRINT ELSE 12200
11100 REM
11200 REM ----- START-UP TIME CONSTANT -----
11300 PRINT "THIS WILL TAKE SOME TIME"
11400 GOSUB 9700
11500 GOSUB 37700
11600 PRINT
11700 PRINT
11800 PRINT "FREQUENCY AT WHICH LOOP GAIN = 1 AT 0 DEGREES:"
11900 GOSUB 26600
12000 PRINT : PRINT "THIS YIELDS A START-UP TIME CONSTANT OF ";CSNG(1000000/(2*PI*SQ));" MICROSECS"
12100 GOTO 6800
12200 IF N=5 THEN PRINT ELSE 7300
12300 REM
12400 REM ----- STEADY-STATE ANALYSIS -----
12500 PRINT "STEADY-STATE ANALYSIS"
12600 PRINT
12700 PRINT "SELECT: 1. 8031/8051"
12800 PRINT "          2. 8751"
12900 PRINT "          3. 8035/8039/8040/8048/8049"
13000 PRINT "          4. 8748/8749"
13100 INPUT ICX
13200 IF ICX<1 OR ICX>4 THEN 12600
13300 GOSUB 46900
13400 GOTO 7300
13500 REM SUBROUTINE BELOW DEFINES INPUT-OUTPUT CURVE OF OSCILLATOR CKTRN
13600 IF ICX>2 AND VO=5 AND VI<2 THEN RETURN
13700 VO = -10*VI + 15
13800 IF VO>5 THEN VO = 5
13900 IF VO<2 THEN VO = 2
14000 IF ICX>2 AND VO>2 THEN VO = 5
14100 RETURN
14200 REM
14300 REM *****
14400 REM
14500 REM DEFINE CIRCUIT PARAMETERS
14600 REM
14700 INPUT " R1 (OHMS)";R1
14800 INPUT " L1 (HENRY)";L1
14900 INPUT " C1 (PF)";X
15000 C1 = X*1E-12
15100 INPUT " C0 (PF)";X
15200 C0 = X*1E-12
15300 INPUT " CXTAL1 (PF)";X
15400 CX = X*1E-12
15500 INPUT " CXTAL2 (PF)";X
15600 CY = X*1E-12

```

230659-45


```

15700 INPUT " GAIN FACTOR MAGNITUDE":AV#
15800 INPUT " AMP FEEDBACK RESISTANCE (K-OHMS)".X
15900 RX = X*1000#
16000 INPUT " AMP OUTPUT RESISTANCE (K-OHMS)".X
16100 RO = X*1000#
16200 REM
16300 REM
16400 REM          LIST1 CURRENT PARAMETER VALUES
16500 GOSUB 17100
16600 RETURN
16700 REM
16800 REM
16900 REM *****
17000 REM
17100 REM          LIST CURRENT PARAMETER VALUES
17200 REM
17300 PRINT
17400 PRINT "CURRENT PARAMETER VALUES: 1. R1 = ",R1," OHMS"
17500 PRINT " 2 L1 = ",CSNG(L1)," HENRY"
17600 PRINT " 3 C1 = ",CSNG(C1*1E+12)," PF"
17700 PRINT " 4 CO = ",CSNG(CO*1E+12)," PF"
17800 PRINT " 5 CXTAL1 = ",CSNG(CX*1E+12)," PF"
17900 PRINT " 6 CXTAL2 = ",CSNG(CY*1E+12)," PF"
18000 PRINT " 7. AMPLIFIER GAIN MAGNITUDE = ",AV#
18100 PRINT " 8. FEEDBACK RESISTANCE = ",CSNG(RX* 001)," K-OHMS"
18200 PRINT " 9. OUTPUT RESISTANCE = ",CSNG(RO* 001)," K-OHMS"
18300 PRINT
18400 PRINT "TO CHANGE A PARAMETER VALUE, TYPE (PARAM NO ),(NEW VALUE)."
18500 PRINT "OTHERWISE, TYPE 0.0 "
18600 INPUT NX,X
18700 IF NX=0 THEN RETURN
18800 IF NX=1 THEN R1 = X
18900 IF NX=2 THEN L1 = X
19000 IF NX=3 THEN C1 = X*1E-12
19100 IF NX=4 THEN CO = X*1E-12
19200 IF NX=5 THEN CX = X*1E-12
19300 IF NX=6 THEN CY = X*1E-12
19400 IF NX=7 THEN AV# = X
19500 IF NX=8 THEN RX = X*1000#
19600 IF NX=9 THEN RO = X*1000#
19700 GOTO 17400
19800 REM
19900 REM
20000 REM *****
20100 REM
20200 REM          CIRCUIT ANALYSIS
20300 REM
20400 REM This routine calculates the loop gain at complex frequency SQ+jFQ.
20500 REM
20600 REM 1. Crystal impedance: RE + jXE
20700 REM
20800 X1 = FNXL(FQ,L1) + FNXC(SQ,FQ,C1)
20900 RE = FNRP((R1+FNRL(SQ,L1)+FNRC(SQ,FQ,C1)),X1,FNRC(SQ,FQ,CO),FNXC(SQ,FQ,CO))
21000 XE = FNXP((R1+FNRL(SQ,L1)+FNRC(SQ,FQ,C1)),X1,FNRC(SQ,FQ,CO),FNXC(SQ,FQ,CO))
21100 REM
21200 REM 2. RF + jXF = (RE+jXE):(amplifier feedback resistance)
21300 REM
21400 RF = FNRP(RX,O,RE,XE)
21500 XF = FNXP(RX,O,RE,XE)
21600 REM
21700 REM 3. Input impedance: Zi = RI + jXI = impedance of CXTAL1
21800 REM
21900 RI = FNRC(SQ,FQ,CY)
22000 XI = FNXC(SQ,FQ,CY)
22100 REM
22200 REM 4. Load impedance: ZL = (impedance of CXTAL2):(RF+RI)+j(XF+XI)]
22300 REM
22400 RL = FNRP((RF+RI),(XF+XI),FNRC(SQ,FQ,CY),FNXC(SQ,FQ,CY))
22500 XL = FNXP((RF+RI),(XF+XI),FNRC(SQ,FQ,CY),FNXC(SQ,FQ,CY))
22600 REM
22700 REM 5. Amplifier gain A = -AV*ZL/(ZL+RO)
22800 REM          = A(real) + jA(imaginary)
22900 REM
23000 AR# = -AV#*FNRR(RL,XL,(RO+RL),XL)
23100 AI# = -AV#*FNXR(RL,XL,(RO+RL),XL)
23200 REM
23300 REM 6. Feedback ratio (beta) = (R1+jX1)/(RF+RI)+j(XF+XI)]
23400 REM          = B(real) + jB(imaginary)

```

230659-46

```

23300 REM
23600 BR# = FNRR(RI, XI, (RI+RF), (XI+XF))
23700 BI# = FNXR(RI, XI, (RI+RF), (XI+XF))
23800 REM
23900 REM 7. Amplifier gain in magnitude/phase form: AR+jAI = A at AP degrees
24000 REM
24100 A = FNZM(AR#, AI#)
24200 AP = FNZP(AR#, AI#)
24300 REM
24400 REM 8. (beta) in magnitude/phase form: BR+jBI = B at BP degrees
24500 REM
24600 B = FNZM(BR#, BI#)
24700 BP = FNZP(BR#, BI#)
24800 REM
24900 REM 9. Loop gain: G = (BR+jBI)*(AR+jAI)
25000 REM
25100 REM
25200 GR = FNRM(AR#, AI#, BR#, BI#)
25300 GI = FNXM(AR#, AI#, BR#, BI#)
25400 REM
25500 REM 10. Loop gain in magnitude/phase form: GR+jGI = AL at AQ degrees
25600 REM
25700 AL = FNZM(GR, GI)
25800 AQ = FNZP(GR, GI)
25900 RETURN
26000 REM
26100 REM
26200 REM *****
26300 REM
26400 REM PRINT CIRCUIT ANALYSIS RESULTS
26500 REM
26600 PRINT
26700 PRINT " FREQUENCY = ", SQ, " + j ", FQ, " HZ"
26800 PRINT " XTAL IMPEDANCE = ", FNZM(RE, XE), " OHMS AT ", FNZP(RE, XE), " DEGREES"
26900 PRINT " (RE = ", CSNG(RE), " OHMS)"
27000 PRINT " (XE = ", CSNG(XE), " OHMS)"
27100 PRINT " LOAD IMPEDANCE = ", FNZM(RL, XL), " OHMS AT ", FNZP(RL, XL), " DEGREES"
27200 PRINT " AMPLIFIER GAIN = ", A, " AT ", AP, " DEGREES"
27300 PRINT " FEEDBACK RATIO = ", B, " AT ", BP, " DEGREES"
27400 PRINT " LOOP GAIN = ", AL, " AT ", AQ, " DEGREES"
27500 RETURN
27600 REM
27700 REM
27800 REM *****
27900 REM
28000 REM SEARCH FOR FREQUENCY (S+JF)
28100 REM AT WHICH LOOP GAIN HAS ZERO PHASE ANGLE
28200 REM
28300 REM This routine searches for the frequency at which the imaginary part
28400 REM of the loop gain is zero. The algorithm is as follows:
28500 REM 1. Calculate the sign of the imaginary part of the loop gain (GI).
28600 REM 2. Increment the frequency.
28700 REM 3. Calculate the sign of GI at the incremented frequency.
28800 REM 4. If the sign of GI has not changed, go back to 2.
28900 REM 5. If the sign of GI has changed, and this frequency is within
29000 REM 1Hz of the previous sign-change, exit the routine.
29100 REM 6. Otherwise, divide the frequency increment by -10.
29200 REM 7. Go back to 2.
29300 REM The routine is entered with the starting frequency SQ+jFQ and
29400 REM starting increment DS+jDF already defined by the calling program.
29500 REM In actual use either DS or DF is zero, so the routine searches for
29600 REM a GI=0 point by incrementing either SQ or FQ while holding the other
29700 REM constant. It returns control to the calling program with the
29800 REM incremented part of the frequency being within 1Hz of the actual
29900 REM GI=0 point.
30000 REM
30100 REM 1. CALCULATE THE SIGN OF THE IMAGINARY PART OF THE LOOP GAIN (GI).
30200 REM
30300 GOSUB 20200
30400 GOSUB 26600
30500 IF GI=0 THEN RETURN
30600 SX# = INT(SGN(GI))
30700 IF SX#=-1 THEN DS = -DS
30800 REM (REVERSAL OF DS FOR GI=0 IS FOR THE POLE-SEARCH ROUTINE.)
30900 REM
31000 REM 2. INCREMENT THE FREQUENCY.
31100 REM
31200 SP = SQ

```

84-222525

230659-47

```

31300 FP = FQ
31400 SQ = SQ + DS
31500 FQ = FQ + DF
31600 REM
31700 REM 3 CALCULATE THE SIGN OF G1 AT THE INCREMENTED FREQUENCY.
31800 REM
31900 GOSUB 20200
32000 GOSUB 26600
32100 IF INT(SGN(G1))=0 THEN RETURN
32200 REM
32300 REM 4 IF THE SIGN OF G1 HAS NOT CHANGED, GO BACK TO 2.
32400 REM
32500 IF SXZ+INT(SGN(G1))=0 THEN PRINT ELSE 31400
32600 SXZ = -SXZ
32700 REM
32800 REM 5 IF THE SIGN OF G1 HAS CHANGED, AND IF THIS FREQUENCY IS WITHIN
32900 REM 1HZ OF THE PREVIOUS SIGN-CHANGE, AND IF G1 IS NEGATIVE, THEN
33000 REM EXIT THE ROUTINE (THE ADDITIONAL REQUIREMENT FOR NEGATIVE G1
33100 REM IS FOR THE POLE-SEARCH ROUTINE)
33200 REM
33300 IF ABS(SP-SQ)<1 AND ABS(FP-FQ)<1 AND SXZ=-1 THEN RETURN
33400 REM
33500 REM 6. DIVIDE THE FREQUENCY INCREMENT BY -10.
33600 REM
33700 DS = -DS/10#
33800 DF = -DF/10#
33900 REM
34000 REM 7. GO BACK TO 2
34100 REM
34200 GOTO 31200
34300 REM
34400 REM
34500 REM *****
34600 REM SEARCH FOR POLE FREQUENCY
34700 REM
34800 REM
34900 REM This routine searches for the frequency at which the loop gain = 1
35000 REM at 0 degrees. That frequency is the pole frequency of the closed-
35100 REM loop gain function. The pole frequency is a complex number, SQ+jFQ
35200 REM (Hz). Oscillator start-up ensues if SQ>0. The algorithm is based on
35300 REM the calculated behavior of the phase angle of the loop gain in the
35400 REM region of interest on the complex plane. The locus of points of zero
35500 REM phase angle crosses the j-axis at the oscillation frequency and at
35600 REM some higher frequency. In between these two crossings of the j-axis,
35700 REM the locus lies in Quadrant I of the complex plane, forming an
35800 REM approximate parabola which opens to the left. The basic plan is to
35900 REM follow the locus from where it crosses the j-axis at the oscillation
36000 REM frequency, into Quadrant I, and find the point on that locus where
36100 REM the loop gain has a magnitude of 1. The algorithm is as follows:
36200 REM 1. Find the oscillation frequency, 0+jFQ
36300 REM 2. At this frequency calculate the sign of (AL-1). (AL = magnitude
36400 REM of loop gain.)
36500 REM 3. Increment FQ
36600 REM 4. For this value of FQ, find the value of SQ for which the loop
36700 REM gain has zero phase.
36800 REM 5. For this value of SQ+jFQ, calculate the sign of (AL-1)
36900 REM 6. If the sign of (AL-1) has not changed, go back to 3
37000 REM 7. If the sign of (AL-1) has changed, and this value of FQ is
37100 REM within 1Hz of the previous sign-change, exit the routine.
37200 REM 8. Otherwise, divide the FQ-increment by -10.
37300 REM 9. Go back to 3
37400 REM
37500 REM 1. FIND THE OSCILLATION FREQUENCY, 0+jFQ
37600 REM
37700 GOSUB 9700
37800 GOSUB 30300
37900 REM
38000 REM 2. AT THIS FREQUENCY, CALCULATE THE SIGN OF (AL-1).
38100 REM
38200 SYZ = INT(SGN(AL-1))
38300 IF SYZ=-1 THEN STOP
38400 REM ESTABLISH INITIAL INCREMENTATION VALUE FOR FQ
38500 F1 = FQ
38600 DF = (FA-F1)/10#
38700 GOSUB 30300
38800 DE = (FQ-F1)/10#
38900 DF = 0
39000 FQ = F1

```

```

39100 REM
39200 REM 3. INCREMENT FQ.
39300 REM
39400 FQ = FQ + DE
39500 REM
39600 REM 4. FOR THIS VALUE OF FQ, FIND THE VALUE OF SQ FOR WHICH THE LOOP
39700 REM GAIN HAS ZERO PHASE. (THE ROUTINE WHICH DOES THAT NEEDS DF = 0,
39800 REM SO THAT IT CAN HOLD FQ CONSTANT, AND NEEDS AN INITIAL VALUE FOR
39900 REM DS, WHICH IS ARBITRARILY SET TO DS = 1000.)
40000 REM
40100 DS = 1000#
40200 SQ = 0
40300 GOSUB 30300
40400 IF AL=1! THEN RETURN
40500 REM
40600 REM 5. FOR THIS VALUE OF SQ+JFQ, CALCULATE THE SIGN OF (AL-1).
40700 REM 6. IF THE SIGN OF (AL-1) HAS NOT CHANGED, GO BACK TO 3.
40800 REM
40900 IF SYX+INT(SGN(AL-1))=0 THEN PRINT ELSE 39400
41000 REM
41100 REM 7. IF THE SIGN OF (AL-1) HAS CHANGED, AND THIS VALUE OF FQ IS WITHIN
41200 REM 1HZ OF THE PREVIOUS SIGN-CHANGE, EXIT THE ROUTINE.
41300 REM
41400 IF ABS(F1-FQ)<1 THEN RETURN
41500 REM
41600 REM 8. DIVIDE THE FQ-INCREMENT BY -10.
41700 REM
41800 DE = -DE/10#
41900 F1 = FQ
42000 SYX = -SYX
42100 REM
42200 REM 9. GO BACK TO 3.
42300 REM
42400 GOTO 39400
42500 REM
42600 REM
42700 REM *****
42800 REM
42900 REM
43000 REM
43100 REM STEADY-STATE ANALYSIS
43200 REM The circuit model used in this analysis is similar to the one used
43300 REM in the small-signal analysis, but differs from it in two respects.
43400 REM First, it includes clamping and clipping effects described in the
43500 REM text. Second, the voltage source in the Thevenin equivalent of the
43600 REM amplifier is controlled by the input voltage in accordance with an
43700 REM input-output curve defined elsewhere in the program.
43800 REM The analysis applies a sinusoidal input signal of arbitrary
43900 REM amplitude, at the oscillation frequency, to the XTAL1 pin, then
44000 REM calculates the resulting waveform from the voltage source. Using
44100 REM standard Fourier techniques, the fundamental frequency component of
44200 REM this waveform is extracted. This frequency component is then
44300 REM multiplied by the factor  $1/ZL/(ZL+RD)!$ , and the result is taken to be
44400 REM the signal appearing at the XTAL2 pin. This signal is then
44500 REM multiplied by the feedback ratio (beta), and the result is taken to
44600 REM be the signal appearing at the XTAL1 pin. The algorithm is now
44700 REM repeated using this computed XTAL1 signal as the assumed input
44800 REM sinusoid. Every time the algorithm is repeated, new values appear at
44900 REM XTAL1 and XTAL2, but the values change less and less with each
45000 REM repetition. Eventually they stop changing. This is the steady-state.
45100 REM The algorithm is as follows:
45200 REM 1. Compute approximate oscillation frequency.
45300 REM 2. Call a circuit analysis at this frequency.
45400 REM 3. Find the quiescent levels at XTAL1 and XTAL2 (to establish the
45500 REM beginning DC level at XTAL1).
45600 REM 4. Assume an initial amplitude for the XTAL1 signal.
45700 REM 5. Correct the DC level at XTAL1 for clamping effects, if necessary.
45800 REM 6. Using the appropriate input-output curve, extract a DC level and
45900 REM the fundamental frequency component (multiplying the latter by
46000 REM  $1/ZL/(ZL+RD)!$ ).
46100 REM 7. Clip off the negative portion of this output signal, if the
46200 REM negative peak falls below zero.
46300 REM 8. If this signal, multiplied by (beta), differs from the input
46400 REM amplitude by less than 1mV. or if the algorithm has been repeated
46500 REM 10 times, exit the routine
46600 REM 9. Otherwise, multiply the XTAL2 amplitude by (beta) and feed it
46700 REM back to XTAL1, and go back to 5
46800 REM 1. COMPUTE APPROXIMATE OSCILLATION FREQUENCY.

```

230659-49


```

46900 GOSUB 9700
47000 REM
47100 REM      2. CALL A CIRCUIT ANALYSIS AT THIS FREQUENCY.
47200 GOSUB 20800
47300 PRINT : PRINT "ASSUMED OSCILLATION FREQUENCY:"
47400 GOSUB 26600
47500 PRINT : PRINT
47600 REM
47700 REM      3. FIND QUIESCENT POINT
47800 REM      (At quiescence the voltages at XTAL1 and XTAL2 are equal. This
47900 REM      voltage level is found by trial-and-error, based on the input-
48000 REM      output curve, so that a person can change the input-output curve
48100 REM      as desired without having to re-calculate the quiescent point.)
48200 VI = 0
48300 VB = 1
48400 K1 = 1
48500 VI = VI + VB
48600 GOSUB 13600
48700 IF ABS(V0-VI)<.001 THEN 49200
48800 IF K1+SIGN(V0-VI)=0 THEN 48900 ELSE 48500
48900 K1 = SIGN(V0-VI)
49000 VB = -VB/10
49100 GOTO 48500
49200 VB = VI
49300 PRINT "QUIESCENT POINT = ",VB
49400 REM
49500 REM      4. ASSUME AN INITIAL AMPLITUDE FOR THE XTAL1 SIGNAL.
49600 EI = .01
49700 NRX = 0
49800 REM
49900 REM      5. CORRECT FOR CLAMPING EFFECTS, IF NECESSARY.
50000 REM      (K1 and K2 are curve-fitting parameters for the ROM parts.)
50100 K1 = (2.5-VB)/(3-VB)
50200 K2 = (VB-1.25)/(3-VB)
50300 IF ICX=2 OR ICX=4 THEN IF EI<(VB+.5) THEN EO = VB ELSE EO = EI -.5
50400 IF ICX=1 OR ICX=3 THEN IF EI<(VB+.5) THEN EO = VB ELSE EO = K1*EI+K2
50500 NRX = NRX + 1
50600 REM
50700 REM      6. DERIVE XTAL2 AMPLITUDE
50800 V0 = 0
50900 VC = 0
51000 VS = 0
51100 FOR NX = -25 TO +24
51200 VI = EO - EI*COS(PI+NX/25)
51300 GOSUB 13600
51400 V0 = V0 + V0
51500 VC = VC + V0*COS(PI+NX/25)
51600 VS = VS + V0*SIN(PI+NX/25)
51700 NEXT NX
51800 V0 = V0/50
51900 V1 = SQR(VC^2+VS^2)/25*FNZM(RL,XL)/FNZM(RL+RD,XL)
52000 REM
52100 REM      7. CLIP XTAL2 SIGNAL
52200 IF V0-V1<0 THEN VL = 0 ELSE VL = V0-V1
52300 PRINT : PRINT "XTAL1 SWING = ",EO-EI," TO ",EO+EI
52400 PRINT "XTAL2 SWING = ",VL," TO ",V0+V1
52500 REM
52600 REM      8. TEST FOR TERMINATION
52700 IF ABS(EI-V1*.8)<.001 OR NRX=10 THEN RETURN
52800 REM
52900 REM      9. FEED BACK TO XTAL1 AND REPEAT
53000 EI = V1*.8
53100 GOTO 50300

```

230659-50

1 COMPUTE APPROXIMATE OSCILLATION FREQUENCY

230659-51